TESIS- TE142599

SIMULASI PERGERAKAN EVAKUASI BENCANA TSUNAMI MENGGUNAKAN ALGORITMA BOIDS DAN A STAR

I MADE PASEK MUDHANA
NRP. 2213206715

DOSEN PEMBIMBING

Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng.

Dr. Supeno Mardi Susiki Nugroho, ST., MT.

PROGRAM STUDI MAGISTER
BIDANG KEAHLIAN TELEMATIKA - CIO
JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2015

THESIS- TE142599

MOVEMENT OF THE TSUNAMI EVACUATION SIMULATION USING BOIDS AND A STAR ALGORITHM

I MADE PASEK MUDHANA
NRP. 2213206715

SUPERVISOR

Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng.

Dr. Supeno Mardi Susiki Nugroho, ST., MT.

MASTER PROGRAM
SPESIALIZATION on TELEMATIC - CIO
ELECTRICAL ENGINEERING DEPARTMENT
FACULTY of INDUSTRIAL TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2015

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Teknik (MT)
di
Institut Teknologi Sepuluh Nopember

Oleh :
I Made Pasek Mudhana
2213 206 715

Tanggal Ujian : 15 Januari 2015
Periode Wisuda : Maret 2015

Disetujui Oleh:

1. Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng. (Pembimbing I)
   NIP: 195809161986011001

2. Dr. Supeno Mardi Susiki Nugroho, ST., MT. (Pembimbing II)
   NIP: 19700313199512100l

3. Dr. Surya Sumpeno, ST., M.Sc. (Penguji)
   NIP: 196906131997021003

4. Dr. Adhi Dharma Wibawa, ST., MT. (Penguji)
   NIP: 197605052008121003

5. Dr. Istas Pratomo, ST., MT. (Penguji)
   NIP: 197903252003121001

Direktur Program Pascasarjana

Prof.Dr.Ir. Adi Soeprijanto, MT
NIP. 196404051990021001

SIMULASI PERGERAKAN EVAKUASI BENCANA TSUNAMI MENGGUNAKAN ALGORITMA *BOIDS* DAN *A STAR*

Nama Mahasiswa	:	I Made Pasek Mudhana
NRP	:	2213206715
Pembimbing I	:	Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng.
Pembimbing II	:	Dr. Supeno Mardi Susiki Nugroho, ST., MT.

ABSTRAK

Musibah bencana alam tsunami yang terjadi menerjang Indonesia khususnya Aceh dan sekitarnya yang memakan banyak korban, telah memberikan gambaran perlunya evakuasi dini pada saat terjadi suatu musibah, khususnya Tsunami.

Penelitian ini mensimulasikan pergerakan kerumunan orang untuk bergerak menuju suatu titik evakuasi pada saat terjadinya gempa bumi yang diperkirakan menimbulkan bahaya tsunami. Simulasi ini menentukan jarak terdekat/terpendek dari posisi individu, meminimalkan terjadinya tabrakan dalam menghindari segala hambatan yang ditemui baik hambatan statis maupun dinamis yang ditemui pada saat melewati rute jalan yang telah ditentukan.

Pada penelitian ini, penerapan model kerumunan diselesaikan dengan menggunakan algoritma *boids*, yang didalamnya terdiri dari algoritma *flocking*, *obstancle avoidance*, *collition detection*, dengan ditambahkan dengan algoritma *A Star pathfinding*.

Setelah diadakan simulasi dan penelitian, maka pergerakan kerumunan manusia yang tersebar didalam area evakuasi, dengan menggunakan algoritma boids dan A Star dapat menghindari halangan dinding dan dapat bergerak menuju titik evakuasi tanpa terjebak di area jalur tusnami.Pertambahan jumlah kerumunan membutuhkan selang waktu yang lebih menuju target yang diinginkan.

Kata kunci: Simulasi, Evakuasi, *Boids*, *Flocking*, *Obstacle Avoidance*, *Collision Detection*, *A Star Pathfinding*.

[*Halaman ini sengaja dikosongkan.*]

MOVEMENT OF THE TSUNAMI EVACUATION SIMULATION USING BOIDS AND A STAR ALGORITHM

Name	:	I Made Pasek Mudhana
NRP	:	2213206715
Supervisor	:	Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng.
Co-supervisor	:	Dr. Supeno Mardi Susiki Nugroho, ST., MT.

ABSTRACT

Natural disasters tsunami hit Indonesia in particular Aceh and its surroundings which claimed many victims, has provided an overview the need for early evacuation in the event of a disaster, especially tsunami.

This study simulates the movement of a crowd of people to move towards an evacuation point at the time of the earthquake were estimated pose a danger of tsunami. This simulation determines the shortest distance or shortest of the individual positions, minimizing collisions in avoiding any obstacles encountered both static and dynamic obstacles encountered in as it passes through the predetermined path.

In this study, the application of the crowd solved by using boids algorithm, which involves a series of flocking algorithms, obstancle avoidance, collition detection , to be added to the algorithm A Star pathfinding.

After held simulation and research, the movement of crowd human spread in evacuation areas, using algorithms boids and A Star can avoid obstacles and walls can move towards a point evacuation without being stuck in a crowd path area tsunami. Increase the number of crowd requires more intervals towards the desired target.

Keywords: Simulation, Evacuation, Boids, Flocking, Obstacle Avoidance, Collision Detection, A Star pathfinding.

[*Halaman ini sengaja dikosongkan.*]

KATA PENGANTAR

Segala puja dan puji penulis haturkan kehadapan Ida Sang Hyang Widhi Wasa, karena atas asung kerta wara nugrahaNya penulis dapat menyelasikan tesis dengan judul:

SIMULASI PERGERAKAN EVAKUASI BENCANA TSUNAMI MENGGUNAKAN ALGORITMA BOIDS DAN A STAR

Tesis ini digunakan sebagai salah satu syarat akademis untuk memperoleh gelar Magister Teknik (M.T) di Jurusan Teknik Elektro, Bidang Keahlian Telematika - CIO, Institut Teknologi Sepuluh Nopember Surabaya.

Tentunya masih banyak kekurangan dalam perancangan dan pembuatan buku tesis ini. Oleh karena itu saran dan kritik yang membangun sangat diharapkan. Semoga buku ini dapat memberikan manfaaat bagi para pembaca. Selain itu penulis berharap agar penelitian tesis ini dapat menambah literatur dan memberikan manfaat bagi semuanya dan mahasiswa Jurusan Teknik Elektro pada khususnya.

Surabaya, Januari 2015

PENULIS

[*Halaman ini sengaja dikosongkan.*]

UCAPAN TERIMA KASIH

Puji syukur saya panjatkan kehadapan Ida Sang Hyang Widhi Wasa atas segala atas segala Asung Kerta Wara NugrahaNya. Saya menyadari bahwa terwujudnya tesis ini tidak lepas dari doa, bantuan, bimbingan serta dukungan dari berbagai pihak.

Dengan selesainya tesis ini, perkenankan pula saya mengucapkan terimakasih yang sebesar-besarnya kepada :

1. Rektor ITS Surabaya, Direktur PPS ITS Surabaya, serta Dekan Fakultas Teknologi Industri ITS Surabaya, atas kesempatan dan fasilitas yang diberikan kepada saya untuk mengikuti dan menyelesaikan Program Megister
2. Direktur Program Pasca Sarjana Institut Teknologi Sepuluh Nopember Surabaya, Bapak Prof. Dr. Ir. Adi Soeprijanto, M.T., atas kesempatan yang diberikan kepada saya untuk menjadi mahasiswa Program Megister.
3. Bapak Prof, Dr.Ir.Mauridhi Hery Purnomo, M.Eng, selaku pembimbing atas semua kesabaran, perhatian, motivasi serta bimbingan dan ilmu pengetahuan yang sangat luar biasa bermanfaat kepada saya sehingga tesis ini dapat diselesaikan.
4. Bapak Dr. Supeno Mardi SN, ST. MT. selaku dosen pembimbing yang telah memberikan bimbingan dan dukungan kepada penulis dalam menyelesaikan buku tesis ini.
5. Ketua Jurusan Teknik Elektro, Bapak Dr. Ir. Tri Arief Sardjono, S.T., M.T
6. Ketua Bidang Keahlian Telematika - CIO, Bapak Dr. Surya Sumpeno, ST., M.Sc.
7. Tim penguji tesis, atas semua masukan dan saran yang sangat berguna untuk perbaikan dan pengembangan penelitian penulis.
8. Bapak-Ibu dosen pengajar bidang keahlian Telematika - CIO, atas ilmu, bimbingan, serta perhatian yang diberikan kepada penulis selama ini.
9. Keluarga besarku, Bapak, Ibu, Anak-anak, dan Saudara-saudara, terimakasih banyak atas dukungan, semangat dan doanya selama ini.
10. Seluruh pihak serta teman-teman sejawat yang tidak dapat saya sebutkan satu persatu, tanpa dukungan Bapak, Ibu, Saudara-saudara sekalian penulis tidak mungkin dapat menyelesaikan tesis ini.

Akhir kata semoga buku ini dapat bermanfaat di masa sekarang dan masa mendatang. Sebagai manusia yang tidak luput dari kesalahan, maka penulis mohon maaf apabila ada kekeliruan baik yang sengaja maupun yang tidak sengaja.

Surabaya, Januari 2015

PENULIS

Daftar Isi

Daftar Tabel

[*Halaman ini sengaja dikosongkan.*]

Daftar Gambar

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Proses evakuasi adalah proses yang sangat membutuhkan ketepatan dalam pengalokasian waktu, terutama didalam penanganan kejadian bencana yang bersifat kompleks seperti evakuasi tsunami. Seperti diketahui bersama tsunami adalah terjadinya gelombang besar yang disebabkan oleh gempa bumi yang berpusat di bawah laut. Biasanya gelombang besar tsunami akan menghantam daerah pesisir pantai dengan kekuatan yang cukup dahsyat yang bisa mengakibatkan korban jiwa dan harta benda.

Dalam kondisi seperti ini, perlu dilakukan evakuasi yang bertujuan untuk menyelamatkan penduduk dari bahaya tsunami, dari tempat yang berpotensi terkena tsunami ketempat yang dianggap aman. Dengan evakuasi diharapkan dapat mengurangi atau meminimalisasi jumlah korban jiwa. Perihal yang paling erat hubungannya dengan evakuasi adalah waktu, semakin lama proses evakuasi atau semakin besar waktu evakuasi yang dibutuhkan maka akan semakin banyak jiwa yang terancam. Selain itu proses evakuasi juga dipengaruhi oleh banyaknya penduduk yang harus diselamatkan serta lokasi dimana para penduduk tersebut berada. Jadi semakin banyak jumlah penduduk dan semakin kompleks bentuk denah pemukiman, maka proses evakuasi akan membutuhkan waktu yang lebih lama.

Pada penelitian sebelumnya (Dewi, Meilany, 2012) menerapkan algoritma *flocking* dan *obstacle avoidance* dengan *collision detection* untuk mensimulasikan pergerakan orang di *mall* keluar menuju pintu utama dalam menghindari hambatan statis berupa benda diam di dalam *mall* dan hambatan dinamis berupa benda bidang bergerak dan meminimalkan frekuensi jumlah tabrakan yang terjadi antar pengunjung karena meningkatnya jumlah populasi.

Menindaklanjuti penelitian (Dewi, Meilany, 2012), dengan mengacu sepenuhnya pada penelitian sebelumnya tentang kelebihan algoritma *boids*, pada penelitian ini akan menerapkan algoritma *boids* dengan menggunakan *flocking*, *obstacle avoidance*, *collision detectio*n dengan *A star pathfinding* untuk mensimulasikan pergerakan orang pada saat evakuasi bencana tsunami menuju suatu titik evakuasi dalam menghindari hambatan statis berupa benda diam di dalam perjalanan menuju

titik evakuasi dan hambatan dinamis berupa benda bidang bergerak dan meminimalkan frekuensi jumlah tabrakan yang terjadi antar penduduk/orang karena meningkatnya jumlah populasi.

1.2 Perumusan Masalah

Berdasarkan latar belakang tersebut, maka pada penelitian ini dirumuskan masalah sebagai berikut:

1. Bagaimana mengurangi terjadinya korban tsunami dengan cara mensimulasikan pergerakan evakuasi bencana tsunami dengan terlebih dahulu menentukan rute terdekat dengan *A star pathfinding* dan mensimulasi pergerakan orang ketika berhadapan dengan hambatan bergerak berupa orang lain dengan menggunakan algoritma *boids*.
2. Bagaimana mengurangi terjadinya korban tsunami dengan cara mensimulasikan pergerakan orang yang bergerak dinamis ketika berhadapan dengan hambatan diam berupa dinding, dengan menggunakan *collision detection* (*circle rectangle* dan *circle line*).

1.3 Batasan Masalah

Pada penelitian ini peneliti membatasi masalah yakni :

1. Penelitian ini disimulasikan dengan menggunakan konsep pemrograman 2 dimensi (2D).
2. Simulasi hanya akan dibatasi pada kasus kerumunan orang dengan *flocking* dan *obstacle avoidance* dengan *collision detection* (*circle rectangle* dan *circle line*).
3. Analisis hambatan pada simulasi penelitian ini berupa hambatan statis berupa dinding/tembok/bangunan dan objek statis yang ada di jalanan yang ditempuh serta objek dinamis yang ada di jalan menuju target.
4. Target yang akan dituju pada simulasi ini hanya 1 (satu) yaitu titik evakuasi pertama.
5. Penelitian ini tidak menampilkan *path* alternatif.
6. Jumlah *boids* yang mewakili orang yang akan disimulasikan berjumlah $<=$ 500 populasi.
7. Simulasi kerumunan orang dewasa dan anak-anak memiliki kemampuan atau kebiasaan yang homogen.

8. Simulasi evakuasi ini hanya menggunakan model evakuasi model pejalan kaki.

1.4 Tujuan Penelitian

Penelitian ini bertujuan untuk mengurangi terjadinya korban tsunami dengan memanfaatkan keunggulan algoritma *A star pathfinding* untuk menentukan rute terpendek, dan algoritma *boids* dalam mengorganisasi kerumunan orang.

1.5 Kontribusi dan Manfaat Penelitian

Hasil penelitian ini diharapkan dapat memberikan kontribusi pada pengembangan deteksi tabrakan antar objek statis dan dinamis dan tentang boids atau perilaku sekelompok agen dalam bidang animasi komputer, serta dapat dapat dimanfaatkan sebagai salah satu metode simulasi kerumunan (crowd simulation). Sedangkan dari segi kebencanaan penelitian ini diharapkan dapat bermanfaat sebagai gambaran awal tentang evakuasi tsunami.

[*Halaman ini sengaja dikosongkan.*]

BAB 2

KAJIAN PUSTAKA

2.1 Definisi Bencana dan Tsunami

Undang-undang Nomor 24 Tahun 2007 Tentang Penanggulangan Bencana menyebutkan definisi bencana sebagai berikut:

Bencana adalah peristiwa atau rangkaian peristiwa yang mengancam dan mengganggu kehidupan dan penghidupan masyarakat yang disebabkan, baik oleh faktor alam dan/atau faktor nonalam maupun faktor manusia sehingga mengakibatkan timbulnya korban jiwa manusia, kerusakan lingkungan, kerugian harta benda, dan dampak psikologis.

Definisi tersebut menyebutkan bahwa bencana disebabkan oleh faktor alam, non alam, dan manusia. Oleh karena itu, Undang-Undang Nomor 24 Tahun 2007 tersebut juga mendefinisikan mengenai bencana alam, bencana nonalam, dan bencana sosial.

Tsunami berasal dari bahasa Jepang yang berarti gelombang ombak lautan (*tsu* berarti lautan, *nami* berarti gelombang ombak). Tsunami adalah serangkaian gelombang ombak laut raksasa yang timbul karena adanya pergeseran di dasar laut akibat gempa bumi.

2.2 Evakuasi

Evakuasi adalah perpindahan langsung dan cepat dari orang-orang yang menjauh dari ancaman atau kejadian yang sebenarnya dari bahaya. Contoh berkisar dari evakuasi skala kecil sebuah bangunan karena ancaman bom atau kebakaran sampai pada evakuasi skala besar sebuah distrik karena banjir, penembakan atau mendekati badai. Dalam situasi yang melibatkan bahan-bahan berbahaya atau kontaminasi, pengungsi sebaiknya didekontaminasi sebelum diangkut keluar dari daerah yang terkontaminasi.

Rencana evakuasi darurat dikembangkan untuk memastikan waktu evakuasi teraman dan paling efisien bagi semua penduduk yang diharapkan dari suatu bangunan, kota, atau wilayah. Sebuah tolok ukur kinerja (*benchmark*) ”waktu evakuasi”

untuk bahaya yang berbeda dan kondisi dibuat. *Benchmark* ini dapat dilakukan melalui penggunaan praktik terbaik, peraturan atau menggunakan simulasi, seperti model aliran manusia dalam sebuah bangunan, untuk menentukan *benchmark*. Perencanaan yang tepat akan menggunakan beberapa jalan keluar serta teknologi untuk memastikan evakuasi penuh dan lengkap. Pertimbangan untuk sejumlah situasi pribadi yang mungkin mempengaruhi kemampuan individu melakukan evakuasi. Situasi-situasi pribadi itu mungkin termasuk sinyal alarm yang menggunakan tanda atau sinyal yang bisa didengar dan dilihat.

Peraturan-peraturan seperti kode bangunan dapat digunakan untuk mengurangi kemungkinan panik dengan memungkinkan individu menyiapkan kebutuhan untuk mengevakuasi diri tanpa menyebabkan alarm. Perencanaan yang tepat akan menerapkan pendekatan semua-bahaya sehingga rencana itu dapat digunakan kembali untuk beberapa bahaya yang mungkin ada. (Abraham, 1994).

2.3 Perilaku Orang Berjalan

Dalam kehidupan nyata, kecepatan berjalan setiap orang tidaklah sama, tergantung oleh banyak faktor, antara lain: umur, jenis kelamin, waktu berjalan (pagi, siang atau malam), tujuan perjalanan, reaksi terhadap environment sekitar, temperatur udara dan lain-lain. Beberapa pakar transportasi menggunakan kecepatan rata-rata 1,20 m/detik (72 m/menit), namun untuk pejalan kaki yang cenderung berjalan lebih lambat, menggunakan kecepatan 0,90 s/d l,00m/detik (54-60m/menit) (Aspelin, 2005).

2.4 Algoritma *Boids*

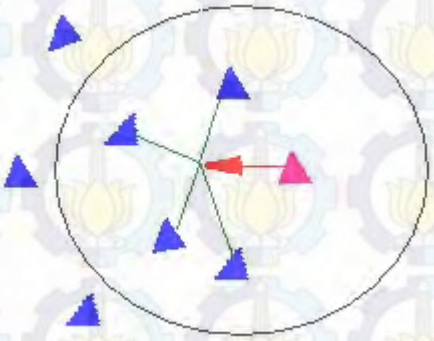
Boids adalah sebuah algoritma yang merepresentasikan gerak dari sebuah kawanan. Perilaku yang dihasilkan sangat mirip dengan kumpulan ikan atau kawanan burung. Gerak boids dihasilkan dari tiga aturan sederhana yaitu *cohesion*, *alignment*, *separation* (Reynolds, 2010).

2.4.1 *Separation*

Pembatasan jika sebuah agen terlalu dekat dengan agen lainnya, dengan cara melakukan penyesuaian arah dan kecepatan untuk menghindari benturan (*collision*). Agen akan menjaga jarak agar tidak nempel dengan *flocksmate* atau tetang-

ganya. Aturan ini mengarahkan (*steering*) agar *Boids* bergerak menghindari kondisi yang padat (*crowded*) oleh kawanan tetangganya. Hal ini memungkinkan B*oids*:

1. Menghindari terjadinya tabrakan
2. Menjaga agar *boids* tetap terpisah pada jarak pisah tertentu yang realistis atau tidak terlalu berdekatan



Gambar 2.1: *Separation* (Reyolds, 2010)

Separation dapat di rumuskan sesuai pada persamaan berikut, (Cui, 2006).

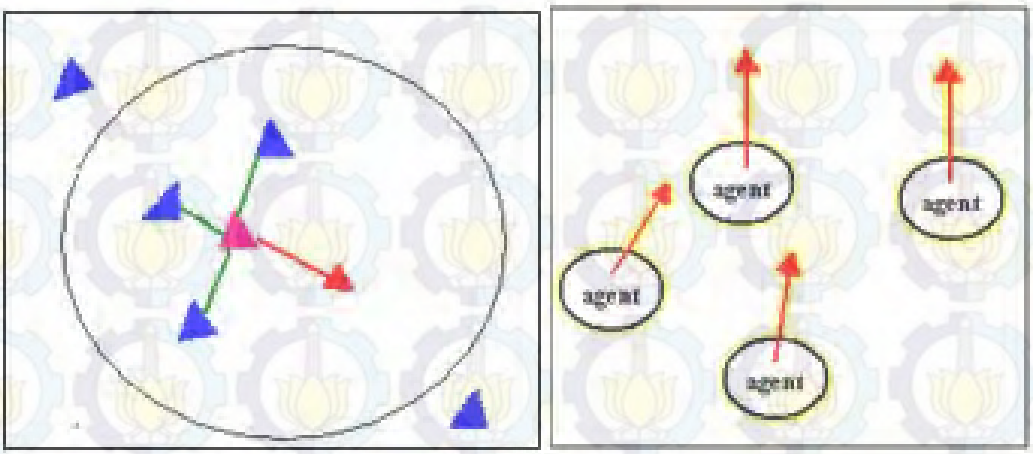
$$D_s = d\left(P_{x'}P_b\right) < d_2 \Rightarrow \overrightarrow{V_{sr}} = \sum_{x}^{n} \frac{\overrightarrow{V_x} + \overrightarrow{V_b}}{d\left(P_{x'}P_b\right)} \tag{2.1}$$

Dimana Vsr adalah kecepatan yang ditentukan oleh aturan separation seperti pada gambar 2.1, d(Px,Pb) adalah jarak antara *boid* x dengan tetangga b, Vx dan Vb adalah kecepatan *boid* x dan *boid* b, d2 adalah nilai jarak yang telah ditetapkan.

2.4.2 *Alignment*

Mengambil rata-rata dari semua percepatan agen yang lain dan melakukan penyesuaian percepatan untuk pindah kearah kelompok. Mengarahkan *agent* menuju posisi rata-rata tetangga. Aturan ini mengarahkan (*steering*) agar *Boids* bergerak ke arah yang merupakan tujuan dari sebagian besar kawanan di tetangga lokalnya. *Boids* berusaha untuk menyesuaikan kecepatannya (arah, kecepatan bergerak) dengan kecepatan tetangga-tetangganya. Hal ini memungkinkan *Boids*:

1. Mengimbangi pemisahan
2. Membuat boids bergerak pada satu arah tujuan umum yang sama



Gambar 2.2: *Alignment* (Reynolds, 2010)

Aligment dapat di rumuskan sesuai pada persamaan berikut, (Cui, 2006).

$$D_a = d\,(P_{x'}P_b) \leq d_1 \bigcap (P_{x'}P_b) \geqslant d_2 \Rightarrow \overrightarrow{V_{ar}} = \frac{1}{n}\sum_{x}^{n} \overrightarrow{V_x} \tag{2.2}$$

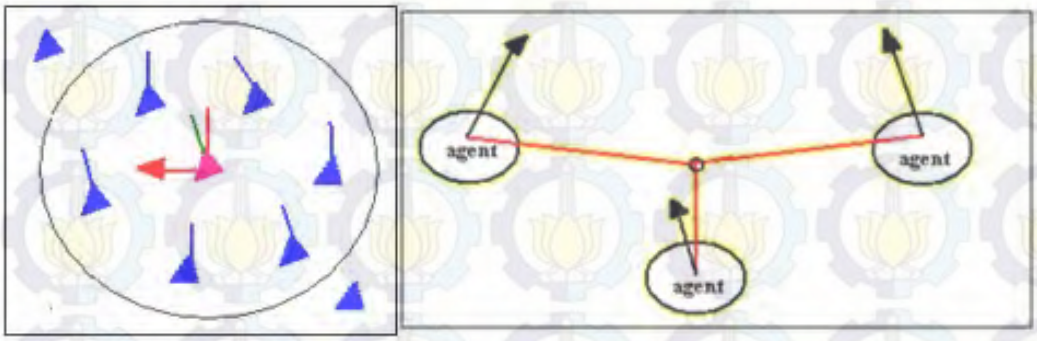
Dimana Var adalah kecepatan yang ditentukan oleh aturan alignment seperti pada gambar 2, d(Px,Pb) adalah jarak antara *boid* x dengan tetangga b, n adalah total jumlah tetangga, Vx adalah kecepatan *boid* x, d1 dan d2 adalah nilai jarak yang telah ditetapkan.

2.4.3 *Cohesion*

Menghitung pusat keseluruhan kelompok dan mengarahkan agen ke arah titik pusatnya. Agen akan mencoba untuk tetap dekat dengan kelompoknya. Aturan ini mengarahkan (*steering*) agar *boids* (agen) bergerak maju ke arah yang merupakan tujuan dan sebagian besar kawanan di tetangga lokalnya. Hal ini memungkinkan *Boids*:

1. Tetap bersama-sama dengan kawanan lokalnya

2. Melakukan kegiatan pengumpulan beberapa kawanan maupun pemisahan kawanan ke dalam 2 kelompok



Gambar 2.3: *Cohesion* (Reynolds, 2010)

Menghitung pusat keseluruhan kelompok dan mengarahkan *agent* ke arah titik pusatnya. Yakni, jumlah posisi dari sernua tetangga dibagi dengan posisi dengan jumlah tetangga. Akhirnya, menggunakan perilaku *seek* agen bergerak menuju titik pusat.

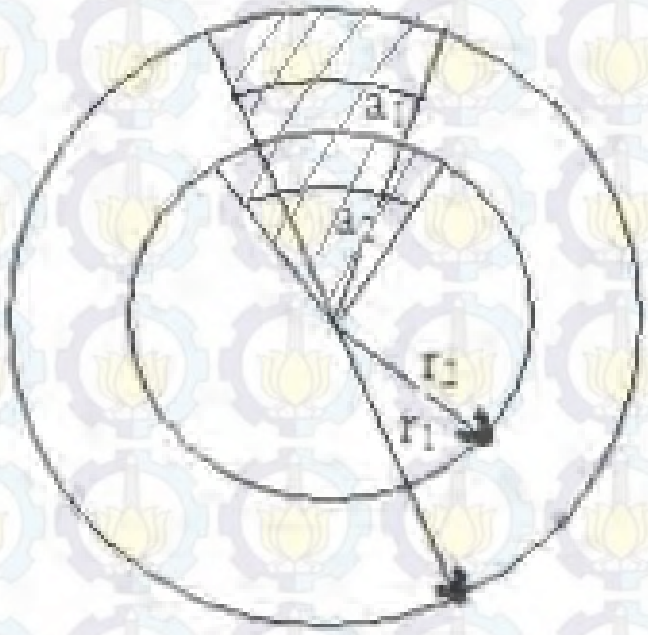
Cohesion dapat di rumuskan sesuai pada persamaan berikut, (Cui, 2006).

$$D_c = d\,(P_x P_b) \leq d_1 \bigcap (P_x P_b) \geqslant d_2 \Rightarrow \overrightarrow{V_{cr}} = \sum_{x}^{n} \overrightarrow{(P_x - P_b)} \tag{2.3}$$

Dimana Vcr adalah kecepatan yang ditentukan oleh aturan *cohesion* seperti pada gambar 3, d(Px,Pb) adalah jarak antara *boid* x dengan tetangga b, d2 adalah nilai jarak yang telah ditetapkan, sedangkan adalah perhitungan arah *vector point*.

2.4.4 *Flocking*

Flocking adalah suatu teknik yang sangat terkenal untuk mensimulasikan perilaku sekawanan entitas atau individu. Merupakan bagian dari *complex steering behaviours*, yang secara umum lebih dikenal dengan istilah *boids"* dengan menggunakan tiga *simple steering behaviours*, yaitu *separation*, *alignment* dan *cohesion*. *Flocking* dikenalkan pertama kali oleh Craig Reynolds pada tahun 1987 pada penelitian nya di SIGGRAPH yang berjudul , *"Flock, Herd, and Schools: .4 Distributed Behavioral Model"* (Reynolds, 2010).



Gambar 2.4: *Neighborhood* dari suatu *agent*

Agen diwakili oleh panah, sedangkan daerah yang diarsir merupakan wilayah persepsi dari agen. Dua *environment* yang berbeda direpresentasikan sebagai agen tunggal yang dapat memiliki *environment* yang berbeda untuk sensor yang berbeda. *Environment* pertama didefinisikan oleh sudut a_1 dan jari-jari r_1, sedangkan *environment* kedua didefinisikan oleh a_2 dan r_2.

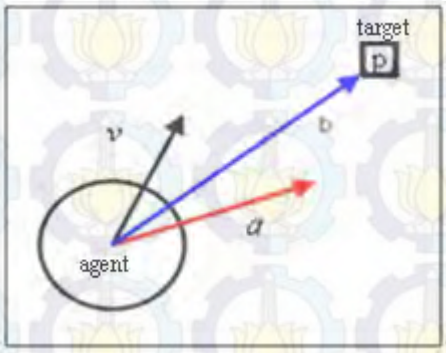
2.4.5 *Steering*

Boids dapat diberi variabel tambahan, seperti kecepatan maksimum dan gaya maksimum kemudi (*steering force*). Yang memiliki metode untuk menghitung vektor kemudi (*steering vector*) menuju lokasi target yang diberikan.
Sebuah vektor yang pada dasarnya merupakan segmen garis terarah yang memiliki arah dan besaran. Dengan grafis, hal ini dapat diterjemahkan sebagai panjang atau jarak dan sudut. Tetapi besarnya juga dapat merupakan faktor- faktor lain seperti kekuatan atau kecepatan obyek. Sebuah skalar, sebagai lawan vektor, hanya memiliki arah besaran (PIGE, 2001).

$$SteeringVector = Vektor yang diinginkan - Velocity \quad (2.4)$$

2.4.6 *Seek*

Perilaku mencari menyebabkan *agent* untuk bergerak mengarah ke arah target dapat dilihat pada Gambar berikut;

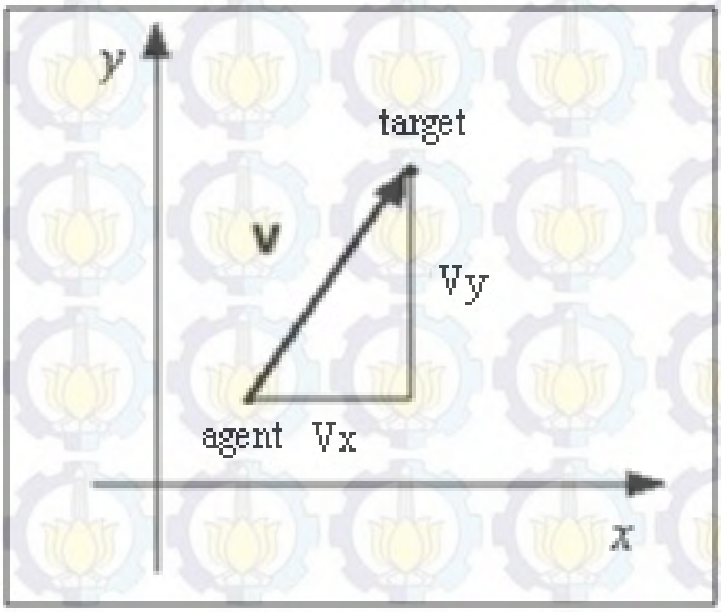


Gambar 2.5: *Seek*

Rumus untuk menjumlahkan vector a :

$$a = Normalize\,(target_p - agent_p)\,x\,agent_{maxspeed} - v \qquad (2.5)$$

Menentukan nilai normalisasi vektor dapat diilustrasikan seperti pada Gambar berikut;



Gambar 2.6: Representasi Vektor

$$V = target - agent$$

$$V = (X_2 - X_1, Y_2 - Y_1)$$

$$V = (V_x, V_y) \tag{2.6}$$

Besar atau panjang vektor dapat ditemukan dengan mengambil akar kuadrat dari jumlah kuadrat masing- masing komponen vektor tersebut.

$$|V| = \sqrt{\left(V_x^2 + Y_y^2\right)} \tag{2.7}$$

Untuk menormalkan vektor, masing- masing komponen vektor d ibagi dengan besarnya vektor. Ketika semua komponen vektor ditambahkan bersama-sama, mereka akan sama dengan 1.

$$V_x = \frac{V_x}{|V|}; V_y = \frac{V_y}{|V|} \tag{2.8}$$

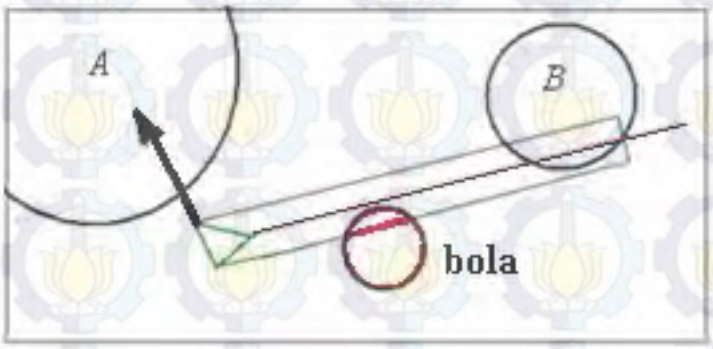
$$V_x + V_y = 1 \tag{2.9}$$

2.5 *Obstacle Avoidance*

Perilaku menghindari hambatan (*obstacle avoidance*) memberikan kemampuan karakter untuk manuver di *environment* dengan menghindari hambatan sekitarnya. Ada perbedaan penting antara menghindari hambatan (*obstacle avoidance*) dan perilaku melarikan diri (*flee*). *Flee* akan selalu mengarahkan karakter untuk menjauh dari lokasi tertentu, sedangkan o*bstacle avoidance* tindakan akan diambil hanya jika suatu hambatan yang terdekat terletak tepat di depan karakter. Sebagai contoh, jika sebuah mobil mengemudi sejajar dengan dinding, *obstacle avoidance* akan mengambil tindakan korektif kemudi, tapi *flee* akan berusaha untuk berpaling dari dinding, akhinya mengemudi tegak lurus dengan dinding. Implementasi dari perilaku *obstacle avoidance* berhubungan dengan penghindaran rintangan dimana tidak harus terjadi tabrakan. Bayangkan sebuah pesawat berusaha untuk menghindari gunung(Dewi, Meilany, 2012).

Tujuan dari perilaku *obstacle avoidance* adalah untuk menjaga sebuah silinder imajiner (sebagai hambatan) yang berada di depan karakter bola, seperti yang diilustrasikan pada gambar di bawah. Silinder A dan B terletak disepanjang sumbu didepan karakter bola. Perilaku menghindari hambatan mempertimbangkan setiap kendala yang pada gilirannya mungkin menggunakan skema *portioning spasial* un-

tuk menyisihkan jarak agar keluar dari hambatan dan menentukan apakah karakter bola bersinggungan dengan silinder (Reynolds, 2010).



Gambar 2.7: *Obstacle Avoidance* (Reynold 2010)

2.6 *Collision Detection*

Collision detection atau pendeteksian tumbukan adalah proses pengecekan apakah beberapa buah objek spasial saling bertumbuk atau tidak. Jika ternyata ada paling sedikit dua buah objek yang bertumbuk, maka kedua objek tersebut dikatakan saling bertumbukkan pada ruang spasial dua dimensi objek yang bertumbuk berarti objek spasialnya beririsan. Teknik pendeteksian tumbukan bisa dikelompokkan menjadi dua macam yaitu *priori detection* dan *post detection*. *Priori detection* adalah pengecekan tumbukan sebelum tumbukan tersebut terjadi, sedangkan *post detection* adalah pengecekan tumbukan setelah tumbukan tersebut terjadi. (Maulana, 2010)

Pada penelitian ini menerapkan teknik *priori detection* dimana pengecekan tumbukan dilakukan sebelum tumbukan terjadi. Setelah menentukan terjadinya *collision*, kita juga harus menentukan respon apa yang terjadi pada objek yang ditabrak dan yang menabrak.

Collision detection digunakan untuk menentukan posisi dan satu objek dengan objek yang lain sehingga tidak ada objek yang saling menembus dan beririsan. Pendeteksian benturan antara obyek yang satu dengan objek yang lain dapat dilakukan dengan mengidentifikasi perpotongan antara obyek yang satu dengan obyek yang lain.

Beberapa algoritma *collision detection* (Compsci, 2011):

1. *Basics: Simple collision detection*
    - *Rectangle - rectangle*
    - *Circle - circle*
    - *Circle - rectangle*
2. *Intermediate*
    - *Line - line*
    - *Circle - line*
    - *Bounding boxes*
3. *Advanced*
    - *Arbitrary polygonal shapes*
    - *Collision detection* berdasarkan waktu untuk mencegah over lap dan meningkatkan *precision*

Pada penelitian ini, akan menerapkan algoritma *Collision Detection* (*Circle - rectangle* dan *circle - line*).

2.7 Algoritma *Pathfinding*

Tujuan dari algoritma *pathfinding* adalah untuk menemukan jalur terbaik dari *vertex* awal ke *vertex* akhir. Secara umum algoritma *pathfinding* digolongkan menjadi dua jenis (Stuart Russel dan Peter Norvig, 1995), yaitu :

1. Algoritma *Uniformed Search* Algoritma *uniformed search* adalah algoritma yang tidak memiliki keterangan tentang jarak atau biaya dari *path* dan tidak memiliki pertimbangan akan *path* mana yang lebih baik. Yang termasuk dalam algoritma ini adalah algoritma *Breadth-First Search*.

2. Algoritma *Informed Search* Algoritma *informed search* adalah algoritma yang memiliki keterangan tentang jarak atau biaya dari *path* dan memiliki pertimbangan berdasarkan pengetahuan akan *path* mana yang lebih baik. Yang termasuk algoritma ini adalah algoritma Dijkstra dan algoritma A*.

2.8 Algoritma A *Star*

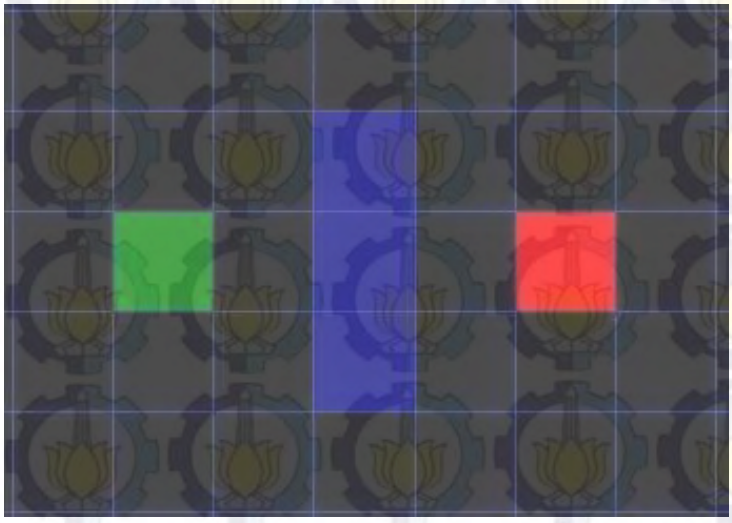
Dalam ilmu komputer, metode A* (A *Star*) adalah *graph search algorithm* yang mencari *path* (jalur) dari titik awal yang diberikan menuju titik tujuan. Algoritma A* pertama kali dijabarkan oleh Peter Hart, Nils Nilsson dan Bertram Raphael pada tahun 1968 (Wikipedia, 2014). Metode A* adalah metode yang merupakan hasil pengembangan dari metode dasar *Best First Search*. Metode ini mengevaluasi setiap titik dengan mengkombinasikan dengan g(n), nilai untuk mencapai titik n dari titik awal, dan h(n), nilai perkiraan untuk mencapai tujuan dari titik n tersebut.

$$f(n) = g(n) + h(n) \tag{2.10}$$

Ketika g(n) memberikan hasil evaluasi nilai untuk mencapai titik n, dan h(n) memberikan nilai estimasi untuk mencapai tujuan dari titik n, maka didapatkan f(n) = nilai estimasi yang terkecil yang melewati titik n.

2.8.1 Cara Kerja Algoritma A *Star* (A*)

Cara kerja algoritma A* dapat digambarkan sebagai berikut, misalkan seseorang ingin berjalan dari *node* A ke *node* B, dimana diantaranya terdapat hambatan, lihat pada gambar. Dimana *node* A ditunjukan dengan kotak berwarna hijau, sedangkan titik B ditunjukkan dengan kotak berwarna merah, dan kotak berwarna biru mewakili hambatan / tembok yang memisahkan kedua *node* tersebut. (Patrick Lester, 2005)



Gambar 2.8: Tampilan Awal Algoritma A *Star*

Perlu diperhatikan bahwa, area pencarian dibagi ke dalam bentuk *node* seperti yang bisa dilihat pada gambar 2.8. Menyederhanakan area pencarian seperti yang telah dilakukan adalah langkah awal dalam pencarian jalur. Dengan fungsi ini dapat menyederhanakan area pencarian menjadi array dua dimensi yang sederhana. Tiap nilai dalam array merepresentasikan satu *node* pada area pencarian, dan status-nya disimpan sebagai yang bisa dilalui atau yang tidak bisa dilalui. Jalur ditemukan dengan menentukan *node* mana saja yang dilalui untuk mencapai *node* B dari *node* A. Ketika jalur ditemukan maka akan berpindah dari satu *node* ke *node* yang lain sampai ke tujuan. Ketika area pencarian sudah disederhanakan ke dalam beberapa *node*. Seperti yang telah dilakukan diatas, langkah berikutnya adalah melakukan pencarian untuk mencari jalur terpendek. Dalam pencarian jalur A*, dimulai dari *node* A, memeriksa *node* yang berdekatan, dan secara umum mencari kesebelah sampai tujuan ditemukan.

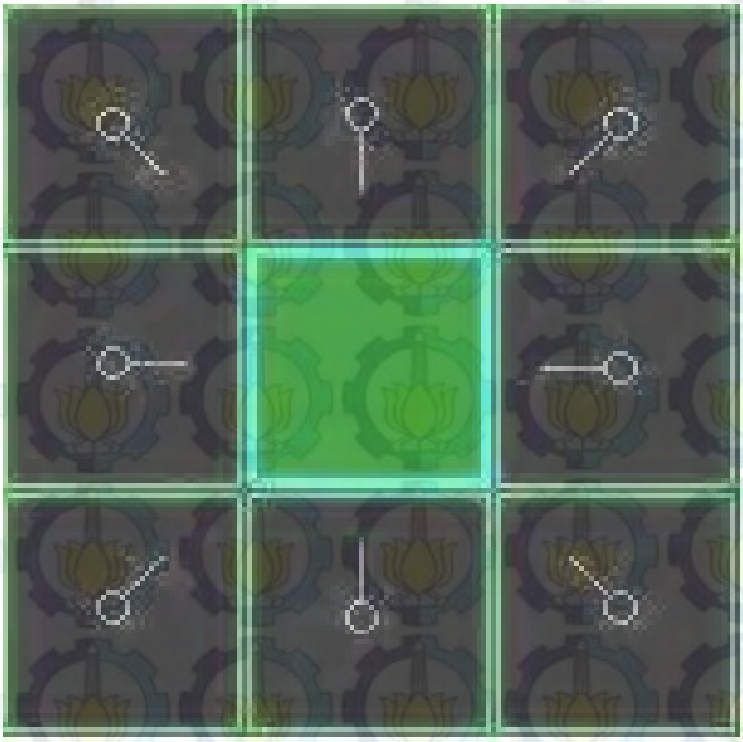
Pencarian dilakukan dengan tahap sebagai berikut :

1. Dimulai dari *start node* A dan *start node* tersebut ditambahkan ke sebuah *open list* dari *node-node* yang akan diperiksa. *List* tersebut berisi *node-node* yang mungkin dilalui pada jalur yang ingin dicari, atau mungkin juga tidak, jadi list tersebut berisi *node-node* yang perlu diperiksa.

2. Lihatlah semua *node-node* yang dapat dilalui yang terhubung dengan *start node*, hindari *node-node* yang merupakan penghalang-penghalang. Tambahkan ke dalam *open list*, untuk tiap-tiap *node*, *node* A merupakan *node parent*, *node* ini berguna ketika ingin mengikuti jalur.

3. Buang *node* A dari *open list*, kemudian tambahkan *node* A ke dalam *closed list*, dimana pada *list* ini tidak perlu lagi memeriksa *node-node* yang ada di dalamnya.

Pada saat ini, harus dilakukan seperti yang terlihat pada gambar 2.9, pada gambar 2.9 *node* yang berwarna hijau di tengah-tengah adalah *start node*. *Node* yang sisinya berwarna biru adalah *node* yang telah dimasukkan ke dalam *closed list*, semua *node* yang bersebelahan dengan *node* pusat yang akan diperiksa dimasukkan ke dalam *open list*, dan sisinya yang berwarna hijau. Tiap petunjuk yang berwarna abu-abu menunjuk ke *node* parent-nya, yang merupakan *start node*.



Gambar 2.9: *Set Parent*

Selanjutnya, dipilih salah satu *node* yang berhubungan dalam *open list* lalu dilakukan berulang-ulang seperti langkah yang akan dijelaskan dibawah ini : Persamaan untuk pemberian nilai pada *node* adalah

$$f(n) = g(n) + h(n) \tag{2.11}$$

Dimana g(n) adalah nilai yang dibutuhkan untuk bergerak dari *start node* A ke sebuah *node* pada area tersebut, mengikuti jalur yang ditentukan untuk menuju kesana. h(n) adalah nilai perkiraan untuk bergerak dari suatu *node* pada area ke *final node* B. *Node* yang dipilih untuk tujuan selanjutnya adalah *node* yang memiliki nilai f(n) terendah.

Jalur yang dibuat adalah jalur yang dibangun secara berulang-ulang dengan menentukan *node-node* yang mempunyai f(n) terendah pada *open list*. Seperti yang telah dikatakan diatas g(n) adalah nilai yang dibutuhkan untuk bergerak dari *start node* ke *final node* dengan menggunakan jalur yang dibuat untuk ke sana. Akan diberi nilai 10 untuk tiap pergerakan *horizontal* atau *vertical*, dan nilai 14 untuk tiap pergerakkan diagonal. Nilai 10 dan 14 digunakan untuk penyederhanaan, di hindari perhitungan decimal dan pengakaran. Cara untuk menentukan nilai g(n) adalah

dengan menghitung nilainya terhadap *node parent*-nya, dengan menambahkan 10 atau 14 tergantung apakah *node* tersebut diagonal atau orthogonal (non-diagonal) terhadap *parent node*. Fungsi ini diperlukan apabila didapatkan suatu *node* berjarak lebih dari *satu node* terhadap *start node*.

h(n) dapat diukur dengan berbagai macam cara. Cara yang digunakan adalah fungsi Manhattan, dimana dihitung jumlah total *node* yang bergerak horizontal atau *vertical* untuk mencapai *final node* dari *node* sekarang, dengan mengacuhkan pergerakkan diagonal. Lalu dikalikan dengan 10. Ini dinamakan fungsi Manhattan karena ini seperti menghitung jumlah blok-blok *node* dari satu tempat ke tempat lain, dimana tidak dapat memotong suatu blok secara diagonal. Yang penting ketika menghitung h(n), harus mengacuhkan rintangan apapun seperti tembok, air, dll. Ini adalah perhitungan perkiraan, bukan jarak nyatanya.

Lalu hitung nilai f(n) dengan menambahkan g(n) dan h(n). Hasilnya dapat dilihat pada gambar 2.10, dimana nilai f(n) ditulis di kiri atas, g(n) kiri bawah dan h(n) di kanan bawah pada tiap-tiap *node*.

Gambar 2.10: Masuk *Close List*

Diberi nilai g(n) sama dengan 10, pada *node* bagian atas, bawah, kiri dan kanan sedangkan *node-node* diagonal diberi nilai 14, karena *node-node* tersebut bersebelahan dengan *start node*.

Nilai h(n) ditentukan dengan menggunakan fungsi manhattan, dimana ditentukan jarak antara *node* tersebut dengan *final node* (merah), dengan bergerak

hanya secara *horizontal* dan *vertical*.

Untuk melanjutkan pencarian, dipilih *node* yang nilai f(n)-nya paling rendah dalam *open list*, ketika dipilih *node* tersebut lalu dilakukan :

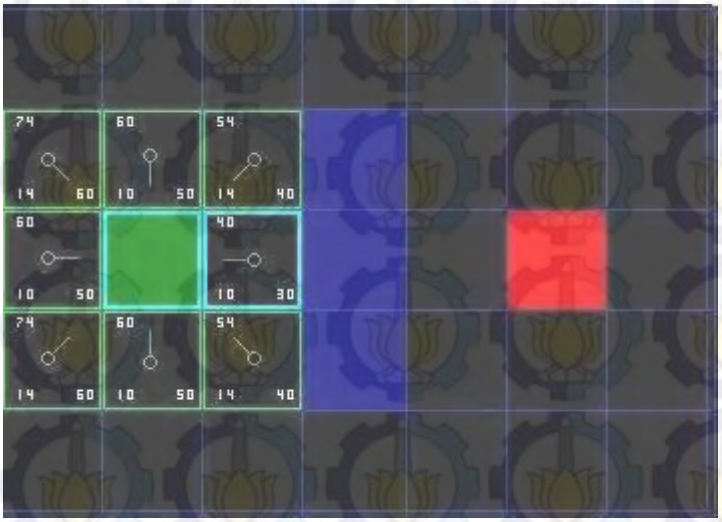
1. Keluarkan *node* tersebut dari *open list* lalu dimasukkan ke *close list*.

2. Periksa semua *node* yang berhubungan. Kecuali *node* yang sudah masuk ke *close list* atau *node* yang tidak dapat dilalui (dinding, air, dan lain-lain). Tambahkan *node* tersebut ke *open list*, apabila *node* tersebut belum dimasukkan ke *open list* tersebut. Jadikan *node* yang dipilih tadi sebagai *parent node* bagi *node* baru tersebut.

3. Jika *node* yang terhubung sudah masuk ke *open list*, periksa apakah nilai g(n) *node* tersebut lebih kecil.

4. Jika tidak jangan lakukan apa-apa, jika benar *parent node* harus diganti lalu dihitung ulang nilai f(n) dan g(n).

Seperti contoh pada gambar 2.10, ada sembilan *node*, dimana 8 *node* masuk *open list* dan *start node* sudah masuk *close list*, lalu *node* dengan nilai f(n) terendah yaitu 40, dimasukkan ke *close list*, karena itu diberi warna biru pada sisinya.

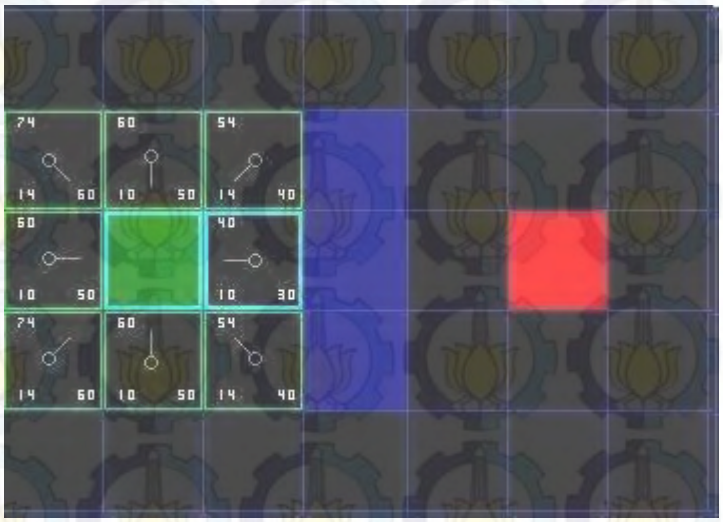


Gambar 2.11: Pemilihan *Close List*

Semua *node* yang berhubungan dengan *node* tersebut diperiksa, *start node* tidak dianggap karena sudah masuk ke *close list*, dan *node* hambatan. 4 *node* lain yang berhubungan semuanya sudah masuk ke *open list* maka harus diperiksa,

apakah nilai g(n) yang dibutuhkan untuk mencapai *node* tersebut melalui *node* yang dipilih lebih kecil daripada menggunakan *node* lain, ternyata seperti contoh di atas didapatkan bahwa apabila ingin ke bawah atau ke atas dari *node* yang dipilih ternyata membutuhkan g(n) sama dengan 20, sedangkan apabila diambil arah diagonal dari start *node* hanya membutuhkan g(n) sama dengan 14, maka tidak dilakukan apa-apa.

Jadi sekarang yang terdapat pada *open list* hanya tinggal 7 *node*, dicari lagi yang nilai f(n) terendah, ternyata ada 2 *node* yang punya nilai f(n) yang sama, itu tidak masalah, Dapat dipilih yang mana saja, tapi untuk mempercepat dapat dipilih yang terakhir masuk ke *open list*. Jadi pilih yang bawah, maka akan tampak seperti gambar 2.12;

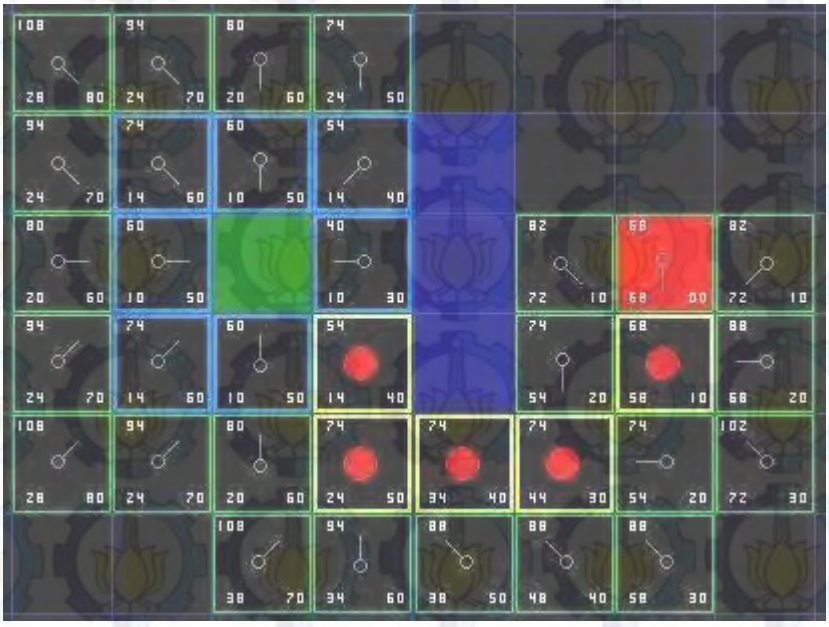


Gambar 2.12: Pemilihan *Close List* 2

Kali ini periksa kembali *node* yang dipilih, masukkan node yang berhubungan ke dalam *open list* kecuali *node* yang merupakan penghalang, node yang sudah masuk *close list* dan *node* yang sudah masuk ke *open list*, tapi disini tidak dapat ditambahkan *node* di bawah dinding ke dalam *open list*, karena tidak dapat langsung dari *node* sekarang ke *node* tersebut tanpa memotong bagian pojok dari dinding di atasnya. Jadi harus turun dulu ke bawah (aturan untuk memotong sudut adalah pilihan). Setelah itu dilakukan proses yang sama seperti yang diatas, lalu ditentukan *node* yang akan dipilih dengan membandingkan nilai f(n). Proses tersebut diulangi sampai *final node* ke dalam *open list* ditambahkan, terlihat pada gambar 2.13;

Gambar 2.13: *Final Node*

Perhatikan pada *node* kedua dibawah *start node*, pada diagram awal nilai g(n) sama dengan 28 dan menunjuk pada *node* kanan atasnya sekarang bernilai g(n) sama dengan 20 dan menunjuk pada *node* di atasnya, hal ini terjadi karena pemeriksaan nilai g(n) dimana nilainya lebih rendah dengan menggunakan jalur yang baru, sehingga *parent node* harus diganti dan nilai g(n) dan f(n) harus dihitung ulang. Lalu setelah selesai ditandai dan proses telah diselesaikan maka ditentukan jalurnya dengan menggunakan fungsi *backtrack* dengan menelusuri dari *final node* mengikuti anak panah pada *node* tersebut hingga sampai ke *start node*. Hasilnya akan terlihat seperti gambar 2.14;



Gambar 2.14: Hasil Akhir Algoritma A *Star*

2.9 Metode dan Asumsi Dasar Simulasi Evakuasi

Menurut Yoso Goto (2011), Simulasi evakuasi orang adalah semacam simulasi aliran kerumunan menggunakan *multiagents*, setiap agen dimodelkan bergerak sepanjang peta jaringan jalan digital mengikuti aturan yang telah ditetapkan. Adapun aturan yang ditetapkan bagi pejalan kaki adalah sebagai berikut;

1. Agen harus mengikuti jaringan jalan link data dari rumah mereka ke lokasi evakuasi mengikuti sesingkat mungkin jalan dalam hal panjang fisik untuk berjalan atau alur waktu terpendek dalam hal waktu yang dibutuhkan
2. Dalam hal kerumunan, agen harus memperlambat, dan pada saat mengalami kemacetan, mereka harus menunggu atau mencari jalan terpendek kedua.
3. Jika jalanan yang dilalui cukup lebar maka agen yang memiliki kecepatan lebih, bisa mendahului agen yang memiliki kecepatan kurang.

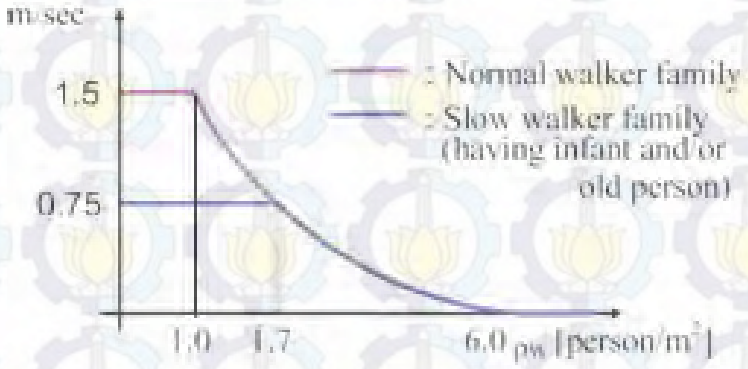
2.9.1 *Agent Modeling*

Dalam penelitiannya, Goso (2011) menjabarkan bahwa orang-orang individu disebut sebagai keluarga. Keluarga seperti yang disebutkan dalam penelitiannya seperti diklasifikasikan ke dalam;

1. Pejalan kaki normal, dalam hal ini orang dewasa.
2. Pejalan kaki yang lambat, dalam hal ini orang tua dan anak-anak.
3. Pengendara sepeda motor.
4. Pengendara mobil.

Dalam evakuasi pejalan kaki dan pengendara mobil, 1 (satu) keluarga dimodelkan sebagai 1 (satu) agen. Dalam evakuasi sepeda motor, 1 (satu) keluarga dipecah menjadi pasangan model, masing-masing pasangan sebagai 1 (satu) agen.

Pejalan kaki yang normal memiliki kecepatan maksimal 5,4 km / jam. Pejalan kaki yang lambat, mewakili keluarga dengan cacat, orang tua, dan anak-anak memiliki kecepatan maksimal 2,7 km / jam. Gambar 2.15 berikut ini, menunjukkan hubungan antara kecepatan berjalan dan kepadatan agen di jalanan. Kecepatan menurun dengan meningkatnya kepadatan, dan menjadi berhenti di kepadatan 6.0 orang/m2. Dalam menghitung kepadatan agen di jalanan, mobil diasumsikan setara dengan 10 pejalan kaki dan sepeda motor untuk 2 pejalan kaki.(Yozo, 2011)



Gambar 2.15: Hubungan kecepatan dengan kepadatan agen

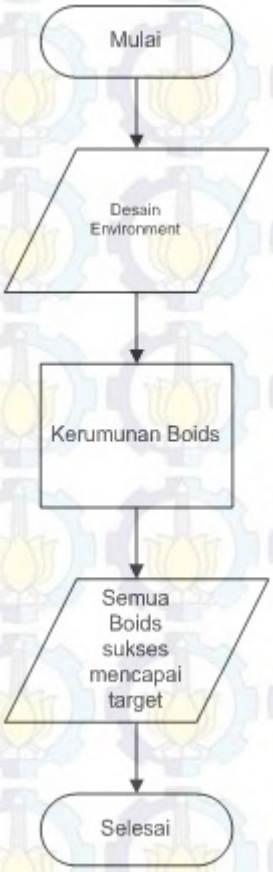
[*Halaman ini sengaja dikosongkan.*]

BAB 3

METODE PENELITIAN

3.1 Pembuatan Model Simulasi Sistem

Pembuatan model simulasi dari algoritma *Boids* mengunakan suatu bahasa pemrograman tertentu, dalam hal ini hanya menggunakan pemodelan dua dimensi. Adapun *flowchart* dari sistem simulasi ini dapat dilihat pada gambar 3.1.

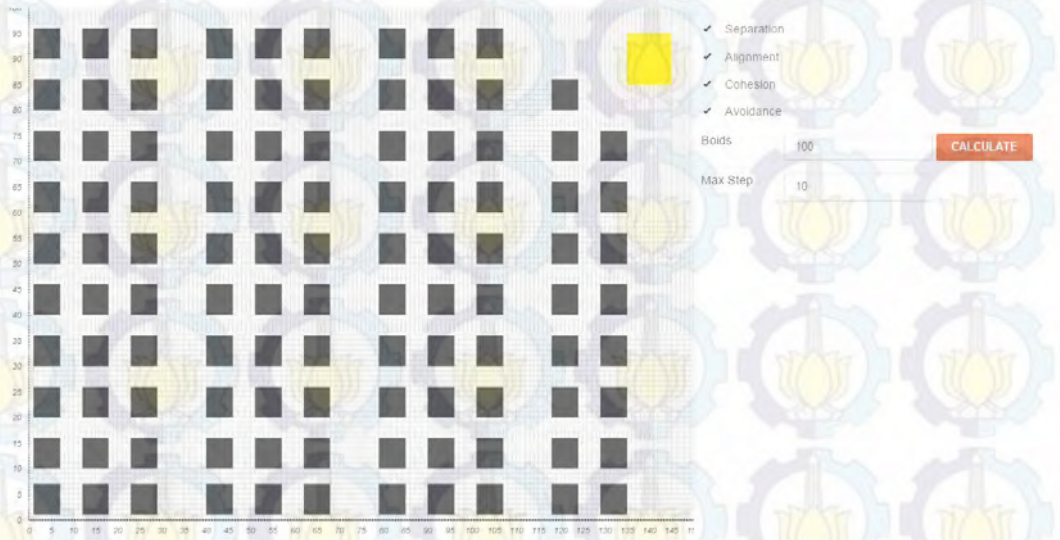


Gambar 3.1: *Flowchart* Sistem Simulasi

Dapat diuraikan dari gambar *flowchart* diatas bahwa pembuatan model simulasi sistem dimulai dari pembuatan desain *Environment*, yang mana selanjutnya dilanjutkan dengan pembuatan kerumunan *boids* yang berupa perancangan algoritma *boids*. Langkah selanjutnya adalah mensimulasikan kerumunan *boids* tersebut sesuai dengan skenario yang diterapkan, dan akan dianggap sukses jika semua *boids* mencapai target yang diinginkan sesuai dengan skenario evakuasi yang diinginkan.

3.2 Desain *Environment*

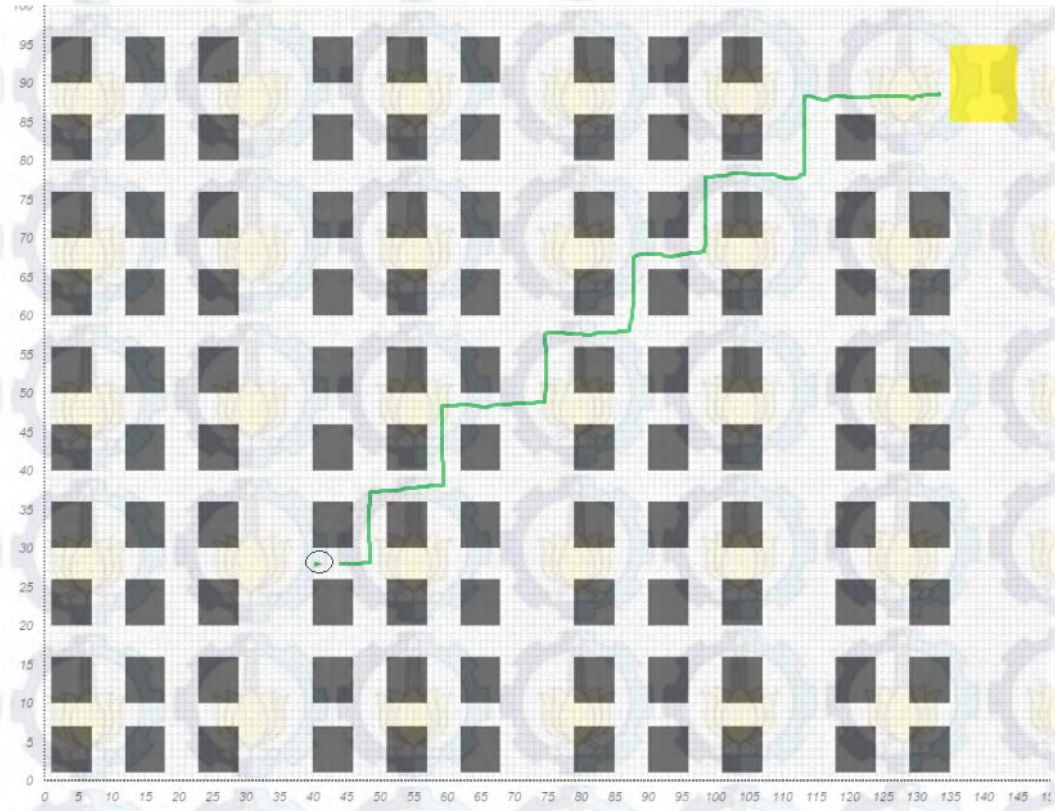
Desain *Environment* terbatas yang dibatasi oleh dinding pembatas pada layar tampilan, sehingga sekelompok orang yang bergerak ke target utama akan menentukan jalur masing-masing dengan terlebih dahulu menentukan jarak terpendek dengan target utama. Setiap orang akan bergerak kearah kerumunan untuk bisa menuju target utama. *Environment* dirancang memiliki satu target utama. Adapun tampilan awalnya dapat dilihat seperti gambar 3.2;



Gambar 3.2: Desain *Environment*

Desain *Environment* dibuat dengan menerapkan kerumunan orang-orang yang bergerak bebas dan acak memiliki target tertentu. Kerumunan dapat dilakukan setelah arah masing-masing kelompok berdasarkan karakteristik diidentifikasi. Volume kerumunan tampak seperti kumpulan partikel *fluida* yang bergerak sesuai dengan kekuatan eksternal yang mendorong mereka (Dewi, Meilany, 2012).

Setiap karakteristik dari kerumunan memiliki tujuan yang sama, pergerakan setiap individu memiliki keputusan yang ditandai dalam kehidupan nyata. Targetnya adalah daerah yang harus diketahui oleh semua orang. Dalam simulasi ini, gerakan target kerumunan menjadi sasaran utama lingkungan seperti yang ditunjukkan pada gambar 3.3:



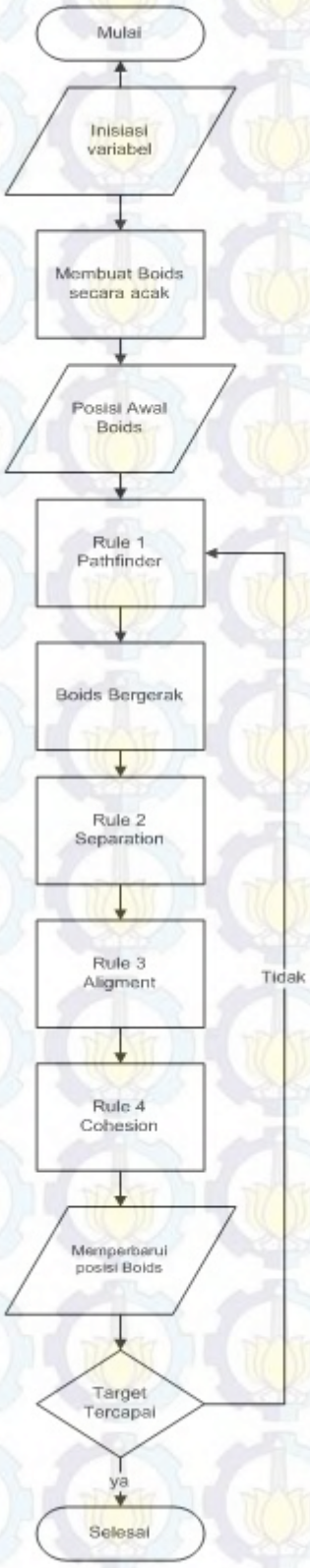
Gambar 3.3: Pergerakan *Boids* menuju target

Dari gambar 3.3, ditunjukkan dengan jelas bahwa agen (*boids*) bergerak menuju target yaitu titik evakuasi yang telah diketahui bersama, dengan terlebih dahulu menentukan *path* atau jalur, yang ditandai dengan garis seperti yang tampak pada gambar 3.3.

3.3 Pembuatan Algoritma *Boids* dengan menggunakan *flocking*, *obstacle avoidance*, *collision detection* dengan *A star pathfinding*.

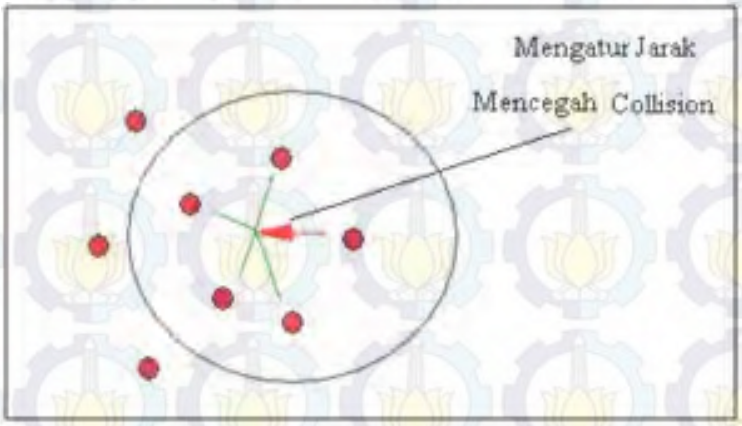
Adapun pembutan Algoritma *boids* dengan menggunakan *flocking*, *obstacle avoidance*, *collision detection* dengan *A star pathfinding* dibuat dengan menerapkan keramaian orang- orang yang bergerak secara bebas dan acak yang mempunyai target tertentu. Kerumunan dapat dibuat dengan mengarahkan kerumunan pada target tertentu dari masing- masing kelompok berdasarkan karakteristik (dewasa dan anak-anak). Volume kerumunan tampak seperti koleksi *fluida* partikel yang bergerak menuju tujuan target utama yang sama. Target adalah area yang harus dituju oleh semua orang. Pada simulasi ini, target pergerakan pengunjung berupa area

titik aman tertentu, dapat dilihat dari *flowchart* pada gambar 3.4 berikut ini;

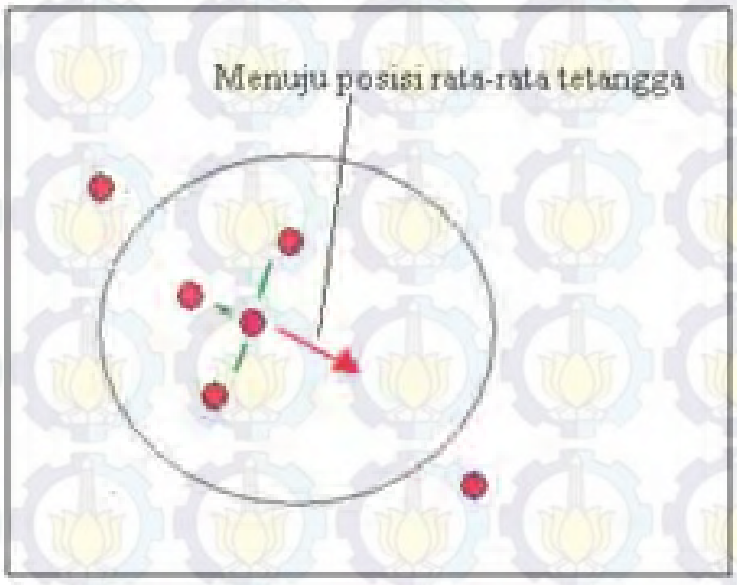


Gambar 3.4: *Flowchart* Algoritma *boids*

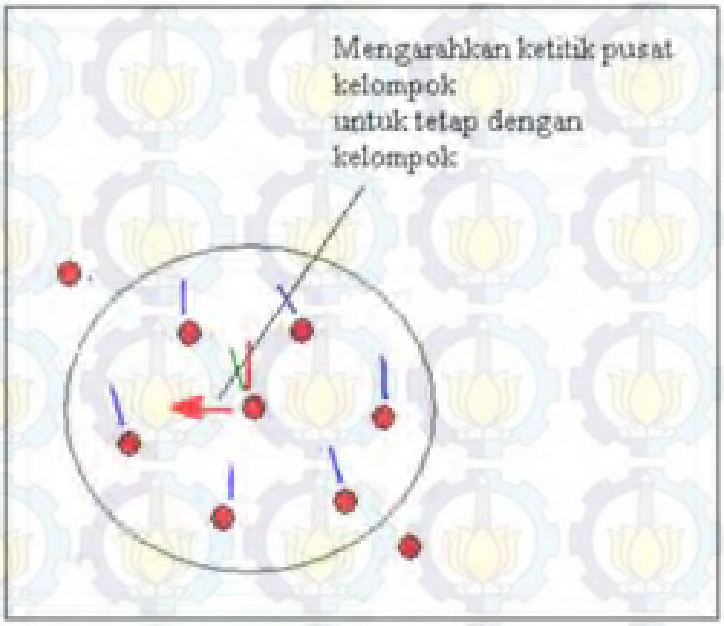
Algoritma *boids* meliputi menjaga jarak agar menghindari terjadinya tabrakan antar agen (tetangga) dalam suatu kelompok tertentu yang mengarahkan *boids* bergerak menghindari kondisi yang padat agar tetap terpisah pada jarak pisah tertentu, *boids* berusaha menyesuaikan kecepatan (arah, kecepatan bergerak) dengan kecepatan agen tetangganya untuk bergerak ke arah tujuan kelompok yang sama dan mengarahkan agen ke titik pusat kelompok untuk tetap dekat dengan kelompoknya dengan mengarahkan *boids* bergerak menuju arah yang merupakan tujuan dari sebagian besar kelompok dapat dilihat pada ilustrasi dibawah ini;



Gambar 3.5: Algoritma *Separation*



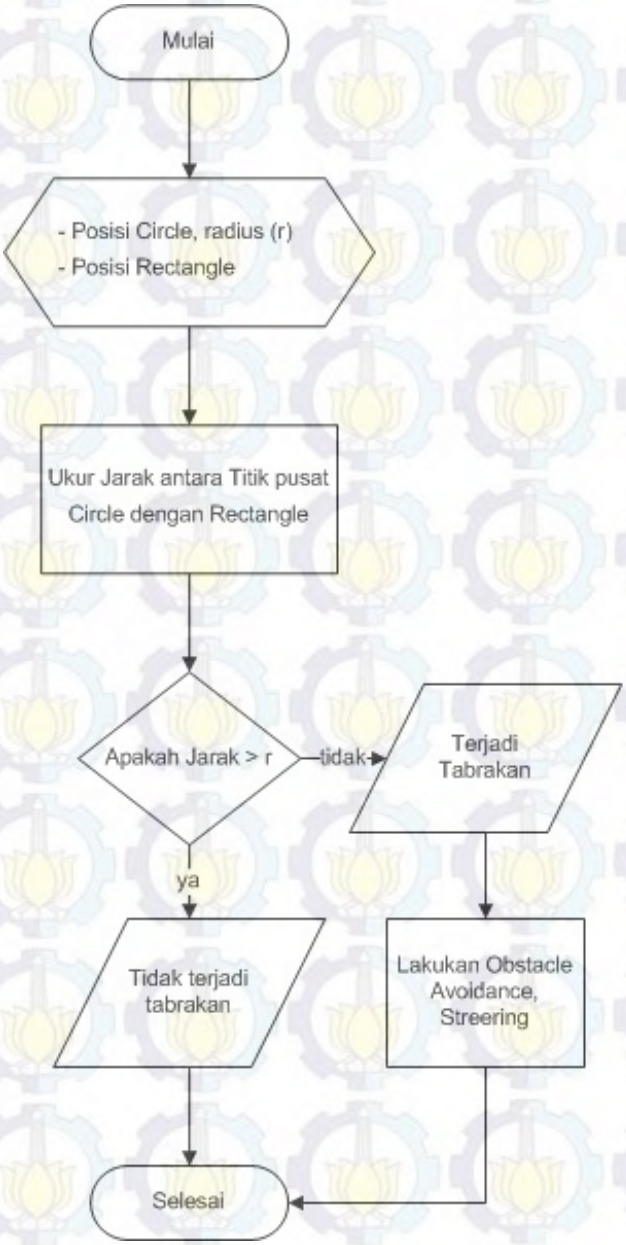
Gambar 3.6: Algoritma *Aligment*



Gambar 3.7: Algoritma *Cohesion*

3.3.1 *Collision Detection* (*Circle Rectangle*)

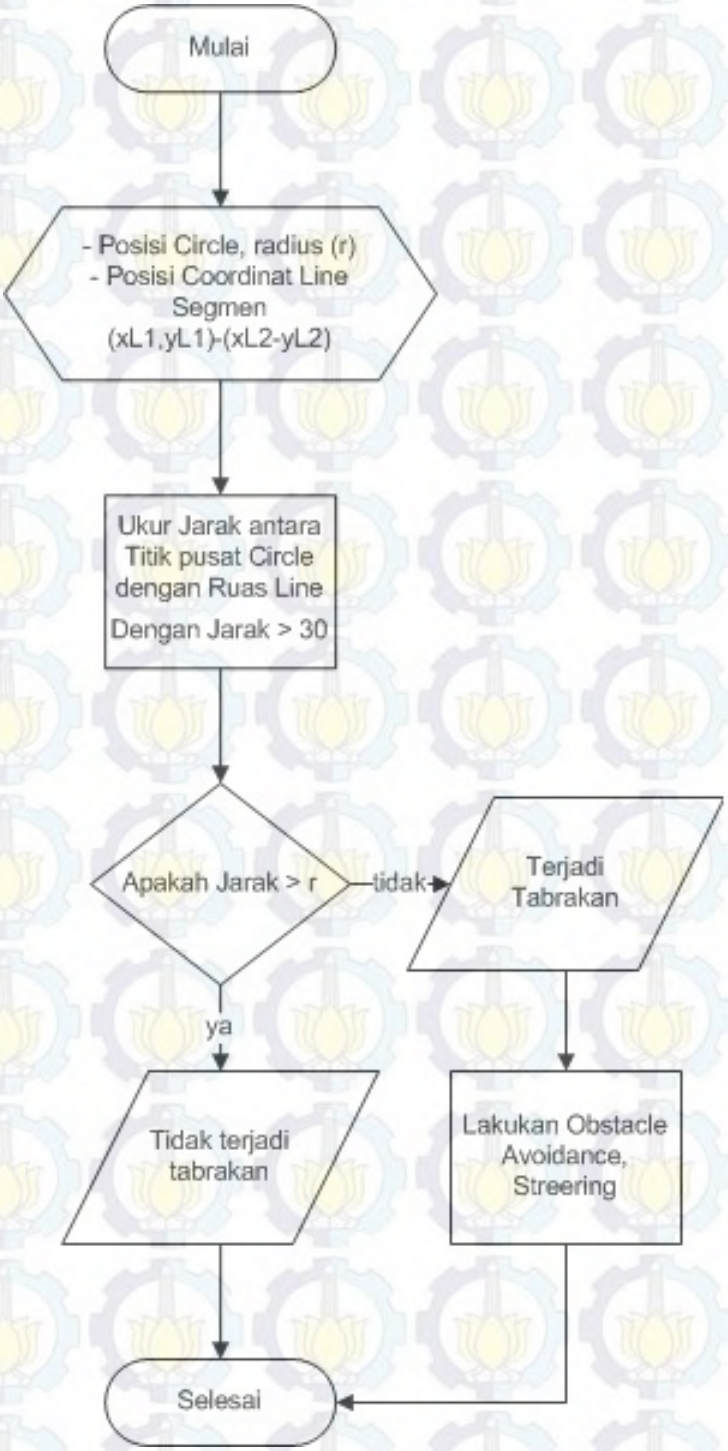
Pergerakan objek (orang) pada kerumunan pengunjung mengikuti aturan *boids* menuju target utama dengan menghindari hambatan bidang statis dan bidang dinamis menggunakan algoritma *Collision Detection* dengan *Circle rectangle*. *Flowchart Collision Detection* dengan *Circle Rectangle* dapat dilihat pada Gambar 3.8 berikut ini;



Gambar 3.8: *Flowchart Collision Detection* dengan *Circle Rectangle*

3.3.2 *Collision Detection* (*Circle Line*)

Selain menggunaka collision detection circle rectangle, simulasi juga menggunakan hambatan collision detection circle line, hambatan dinding di deteksi sebagi line pada sisi bagian dalam dan luar. Flowchart Collision Detection Circle Line dapat dilihat pada Gambar 3.9 berikut.



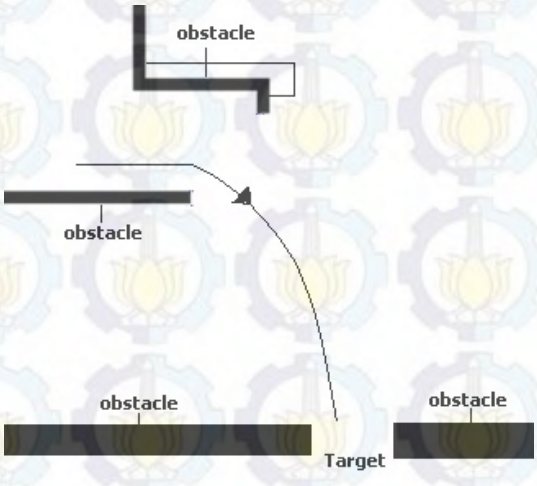
Gambar 3.9: Flowchart Collision Detection Circle Line

Simulasi pendeteksian satu buah hambatan statis oleh sebuah objek untuk mencapai suatu target tertentu dapat diilustrasikan seperti pada Gambar 3.10. Dimana objek akan melakukan deteksi sekitar batas tepi hambatan sebatas minimum jarak yang ditetapkan agar tidak terjadi tabrakan terhadap hambatan tersebut dan

kemudian akan bergerak menghindari hambatan kearah target yang dituju.

Gambar 3.10: menghindari Hambatan Statis

Simulasi untuk menghindari beberapa hambatan statis dapat diilustraikan seperti pada Gambar 3.12. Objek akan mendeteksi beberapa hambatan sekaligus dan mejaga jarak minimal terhadap hambatan, sehingga mencegah terjadinya tabrakan. Selama melakukan deteksi jarak minimal terhadap hambatan, objek akan terus bergerak menuju target yang ingin dicapai hingga target tercapai tanpa terjadi tabrakan terhadap hambatan yang dilaluinya.

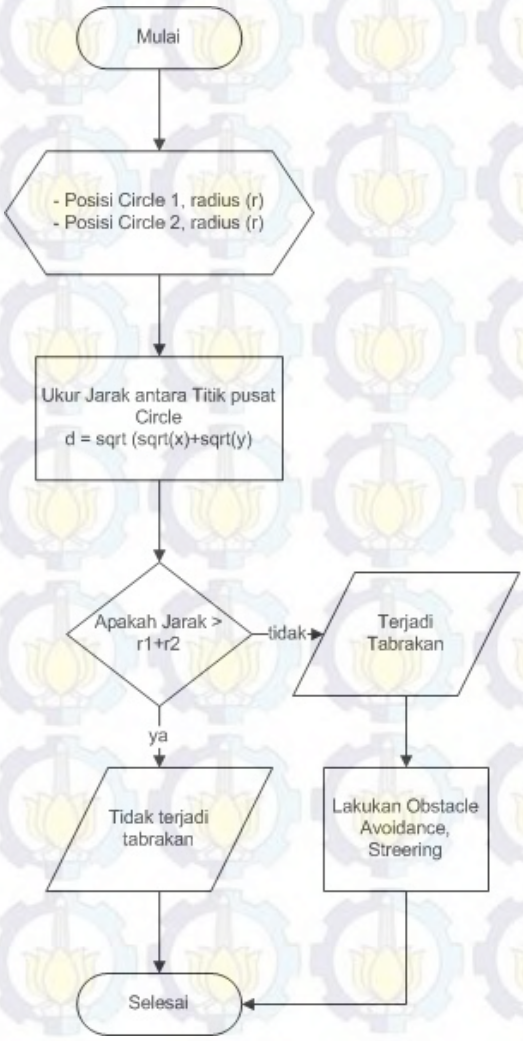


Gambar 3.11: Simulasi menghindari hambatan statis

Bidang objek dinamis berupa line disimulasikan terhadap kerumunan penduduk yang ingin mencapai tujuan target titik evakuasi. Hambatan bergerak dinamis

dan lebih dominan menuju bidang wall sisi sumbu x positif. Hambatan dinamis bergerak searah sumbu x positif dan sumbu x negatif secara terus menerus. Masing-masing penduduk akan bergerak menghindari hambatan dinamis dengan menghitung jarak minimal yang harus dipenuhi agar masing- masing penduduk tidak akan menabrak hambatan tersebut. Penghindaran hambatan akan memberikan aksi perubahan posisi pergerakan penduduk kearah sumbu negatif x (- x), yaitu kesisi ruangan yang lebih luas dan jauh dari bidang hambatan statis lainya.

3.3.3 Collision Detection (Circle Circle)

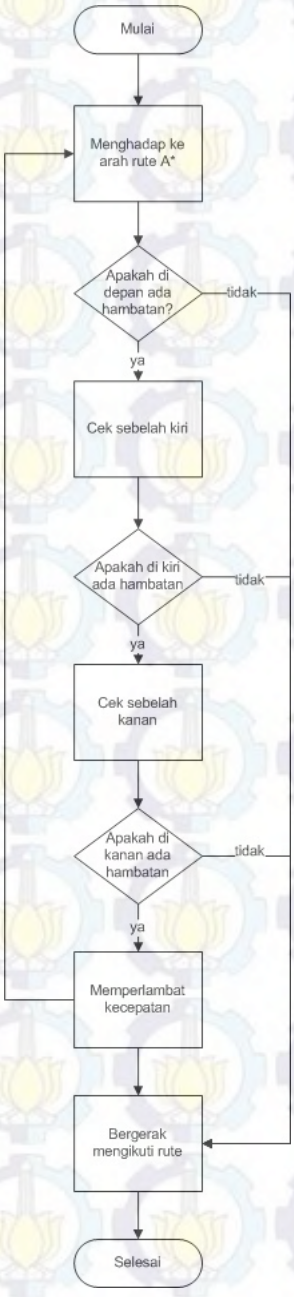


Gambar 3.12: Flowchart Collision Detection dengan Circle Circle

Pergerakan orang pada kerumunan penduduk mengikuti aturan *boids* menuju target titik evakuasi. Namun masih tetap terjadi tabrakan antar orang (*boids*), untuk

mengatasi hal tersebut ditambahkan algoritma *Collision Detection* dengan *Circle circle* pada algoritma *boids* yang ada. Dengan algoritma ini dapat memberikan pemisahan jarak antar orang (*boids*) jika terdeteksi terjadi tabrakan atau overlap. Flowchart *Collision Detection* dengan *Circle Circle* dapat dilihat pada Gambar 3.12.

3.3.4 Obstacle Avoidance



Gambar 3.13: Flowchart *Obstacle Avoidance*

Setelah mendeteksi adanya tabrakan atau overlap, selanjutnya akan dilanjutkan dengan melakukan gerakan penghindaran. Adapun aturan yang berlaku adalah mengikuti aturan algoritma *obstacle avoidance*. Dengan algoritma ini jika terdeteksi adanya hambatan dan terjadi tabrakan, maka dapat dilakukan upaya untuk menghindari tabrakan maupun hambatan yang dimaksud. Flowchart *obstacle avoidance* dapat dilihat pada Gambar 3.13.

3.4 Perancangan Skenario Simulasi Evakuasi

Guna memudahkan pelaksanaan simulasi evakuasi tsunami, perlu dibuatkan skenario jalannya simulasi evakuasi yang diinginkan. adapun skenario evakuasi tsunami secara umum adalah sebagai berikut;

1. Terjadinya gempa bumi yang diperkirakan akan menyebabkan tsunami
2. Terdengar sirene tanda bahaya tsunami
3. Penduduk diasumsikan berada diluar rumah ketika sirene terdengar
4. Penduduk mulai bergerak ke titik evakuasi setelah terdengarnya sirene peringatan dini tsunami.
5. Kerumunan terjadi di jalan menuju titik evakuasi
6. Penduduk mencapai titik evakuasi pada limit waktu tertentu.

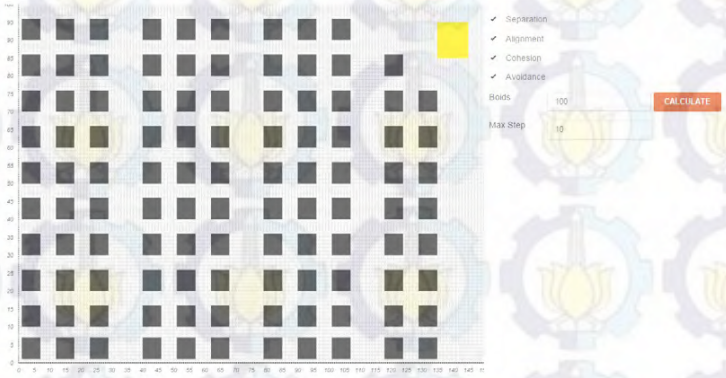
Dalam sistem simulasi evakuasi, posisi agen disimulasikan, dibuat/diatur secara acak (*random*). Posisi setiap agen tidak ada yang sama, tidak bole menempati titik-titik maupun jalur-jalur yang telah ditentukan.

BAB 4

ANALISA HASIL DAN PEMBAHASAN

4.1 Desain Interface Simulasi Sistem

Desain interface simulasi sistem ini diaplikasikan menggunakan bahasa pemrograman HTML dan Java Scrip. Desain dibuat dengan tampilan tanpa pembatas tepi, sehingga terlihat seperti ruangan terbuka. Sedangkan disebelah kiri dan bawah tampilan ditambahan sumbu x dan y sebagi penanda titik atau pixel. Dalam hal penempatan hambatan, pada desain sistem ini, hambatan berupa objek rectangle yang diasumsikan sebagai bangunan yang ditempatkan pada posisi-posisi tertentu dalam bidang yang diingikan. Hambatan tersbut diberi warna hitam sebagai penanda. Sedangkan lapangan tempat berkumpul atuu disebut dengan titik poin evakuasi diberi dengan penanda berwarna kuning.untuk ilustrasinya terlihat pada Gambar 4.1.



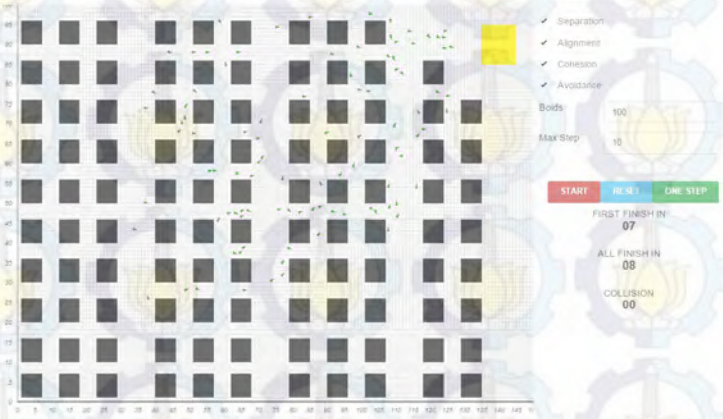
Gambar 4.1: Desain Interface Sistem

4.2 Simulasi Agen Boid

Pada simulasi kerumunan orang di dalam ruangan menggunakan flocking boids dengan mendefinisikan agen Boids sebagai agen yang homogen yaitu agen yang memiliki kemampuan yang sama. Agen ini diwakili oleh boid berbentuk segitiga yang berwarna hijau.

Simulasi kerumunan orang di ruangan terbuka bergerak menuju satu target utama. Kerumunan orang bergerak dengan menghindari hambatan berupa bangunan dan hambatan rectangle statis yang ada di dalam area evakuasi. Kerumunan juga akan bergerak mencapai target utama dengan menjaga jarak antar sesama kerumunan sehingga tidak saling bertabrakan, dan selalu bergerak ke kelompok kerumu-

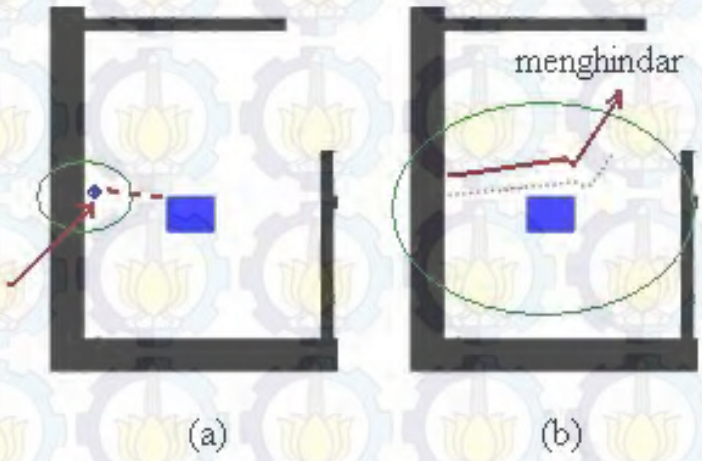
nan untuk menuju target yang sama. Simulasi kerumunan orang dapat digambarkan pada Gambar 4.2.



Gambar 4.2: Simulasi Kerumunan

4.3 Pergerakan orang menghindari Hambatan Statis

Jika seorang warga/penduduk dalam perjalanannya menemui hambatan statis, yakni menabrak dinding bangunan dan hambatan statis berupa bidang rectangle yang ada di tengah-tengah area. Hasil pengujian simulasi dari seorang agen dengan bidang rectangle yang ada di tengah ruangan area adalah sebagai berikut : Dengan posisi hambatan rectangle (100, 350, 140, 380).



Gambar 4.3: Pergerakan orang menghidar hambatan statis. a). Kondisi awal. b). Kondisi Setelah bergerak

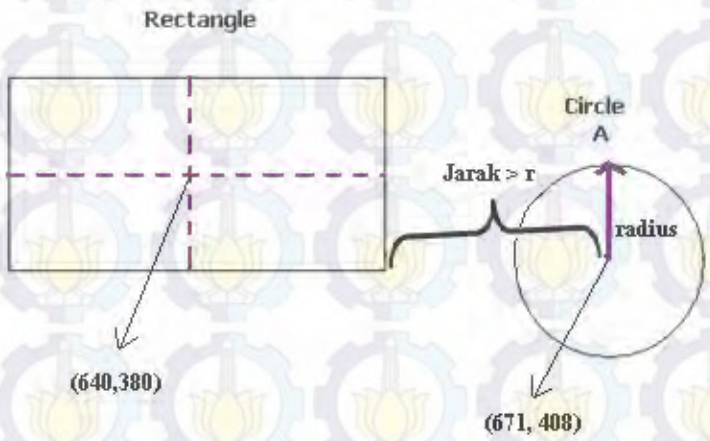
Pada Gambar 4.3 merupakan hasil simulasi untuk jarak orang ke rectangle kurang dari jari-jari bola r. Hasil pengukuran posisi koordinat orang ketika

menghindari hambatan rectangle yang berada pada koordinat (620, 360, 660, 400) diperlihatkan pada Tabel 4.1.

Tabel 4.1: Posisi Orang vs hambatan

No Pergerakan	Posisi Bola	
	X	Y
1	726	484
2	899	463
3	670	443
4	671	408
5	649	351
6	652	316
7	624	294
8	593	278

Pada Gambar 4.4 posisi rectangle untuk sumbu X,Y (620,360) dan posisi lingkaran pada sumbu X,Y (671, 408). Dari kedua posisi objek tersebut dapat dihitung d yaitu jarak anta titik pusat circle dengan permukaan rectangle.



Gambar 4.4: Jarak Rectangle terhadap radius circle

Geometri objek dalam program adalah geometry circle (radius. 4) dan geometry rectangle (40,40).

Sisi kanan rectangle :

$P_0 = (640 + \frac{40}{2})$

$P_0 = 660$

$normal = (660, 0)$

$persamaanplanar = Ax + Bx + C = 0$
$persamaanplanar = 660 + C = 0$
$C = -660$
untuk vektor normal adalah:
$normal = \frac{N}{|N|} = \frac{(A,B)}{\sqrt{(A,B)^2}} = \frac{(660)}{\sqrt{(660)^2}} = 1$
nilai jarak (d):
$d = normal.PC + C$
$d = (1.(671) + (-660)$
$d = 11$
Jadi jarak rectangle dengan circle (orang atau boids) adalah d > r dimana (11> 4), sehingga tidak terjadi tabrakan.

4.4 Pengamatan terhadap pergerakan seseorang jika berpapasan dengan orang lain

Pengamatan dilakukan terhadap pergerakan seseorang jika berpapasan dengan orang lain untuk menuju target utama. Pengamatan dilakukan terhadap simulasi orang sesuai desain sistem.

Tabel 4.2: Data simulasi pergerakan seseorang jika berpapasan dengan orang lain

Simulasi ke-	Jumlah Populasi	Jumlah Tabrakan
1	10	0
2	50	0
3	100	1
4	200	4
5	300	7
6	400	12
7	500	17

Dari data hasil simulasi untuk pergerakan seseorang jika berpapasan dengan orang lain pada Tabel 4.2 diatas, terlihat masih terjadi tabrakan antar orang atau penduduk. Terlihat juga dari data hasil yang diperoleh, jika jumlah populasinya makin bertambah atau makin besar, maka frekuensi terjadinya tabrakan juga akan semakin besar. Tingkat frekuensi tabrakan bisa mencapai 17 untuk jumlah populasi penduduk 500.

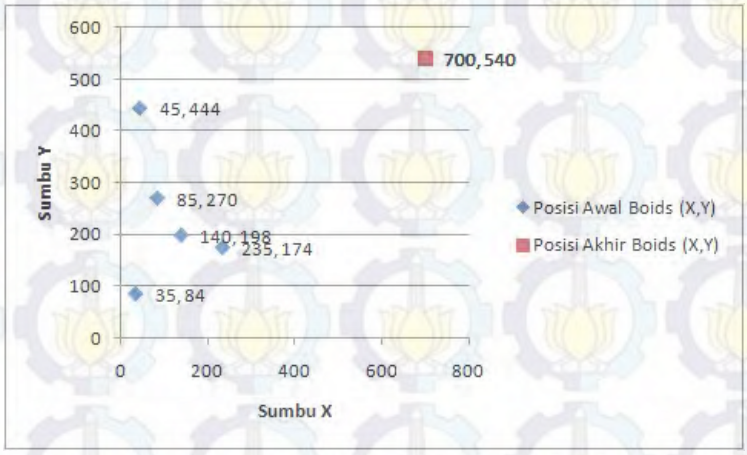
4.5 Pengukuran Pergerakan Agen

Hasil pengukuran pergerakan agen dari posisi awal ke posisi akhir diperlihatkan pada Tabel 4.3.

Tabel 4.3: Pengukuran Pergerakan Agen

No	Posisi awal (x,y)	Posisi akhir (x,y)
1	235,174	700,540
2	35,84	700,540
3	45,444	700,540
4	85,270	700,540
5	240,298	700,540

Pada Tabel 4.3 hasil pengukuran pergerakan agen dari posisi awal menuju posisi target utama berdasarkan koordinat sumbu x dan sumbu y pada simulasi pergerakan dapat ditunjukan pada grafik Gambar 4.5 dengan level posisi awal (xP1, yP1) = (235,174) dan akhir (xP2, yP2) = (700,450).



Gambar 4.5: Pengukuran Pergerakan Agen

4.6 Analisa Pencapaian Hasil Pergerakan Populasi Menuju Target Utama

Analisa pencapain hasil simulasi tentang bagaimana pergerakan kerumunan orang menuju satu target utama diperoleh dengan melakukan uji coba pengaruh waktu yang diperlukan untuk mencapai target utama terhadap jumlah populasi kerumunan. Pada simulasi menggunakan kecepatan (velocity) agen = 1,0 pixel/detik.

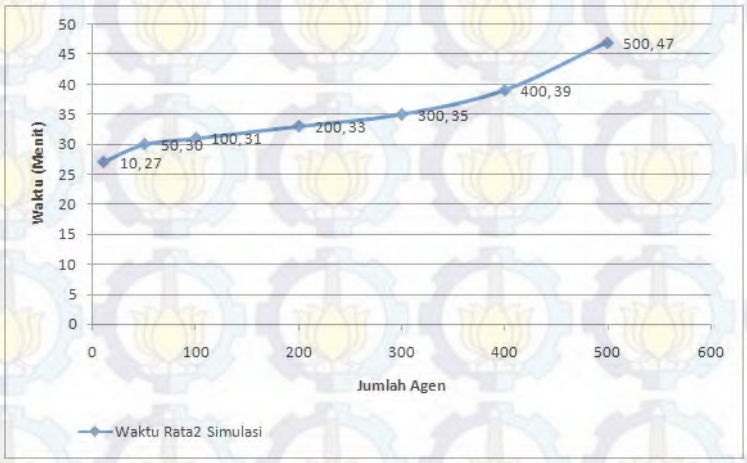
Simulasi dilakukan pada lebar layar : 150 pixel dan tinggi = 100 pixel. Simulasi ini mewakili ruang sebenarnya yang berukuran 900 x 600 m, dengan kecepatan 1.2 m/detik. Data hasil pengukuran pengaruh waktu yang diperlukan untuk mencapai target utama terhadap jumlah populasi kerumunan dapat dilihat pada Tabel 4.5. Terlihat dari hasil tabel tersebut, semua orang dalam berbagai jumlah populasi yang berbeda-beda di dalam area evakuasi, berhasil menuju target utama tanpa ada satupun yang tertinggal. Namun, semakin banyak atau semakin meningkat jumlah populasi kerumunan orang atau penduduk dalam area maka akan semakin banyak waktu yang diperlukan oleh kerumunan tersebut untuk mencapai target utama

Tabel 4.4: Pengukuran Pengaruh Waktu terhadap Jumlah Populasi Kerumunan

Simulasi ke-	Ujicoba ke-	Jml Populasi	Jml sampai	Waktu rata2 (menit)	kecepatan rata-rata
1	1	10	10	27	2.7
2	2	10	10		
3	3	10	10		
4	4	10	10		
5	1	50	50	30	0.6
6	2	50	50		
7	3	50	50		
8	4	50	50		
9	1	100	100	31	0.31
10	2	100	100		
11	3	100	100		
12	4	100	100		
13	1	200	200	33	0.165
14	2	200	200		
15	3	200	200		
16	4	200	200		
17	1	300	300	35	0.116667
18	2	300	300		
19	3	300	300		
20	4	300	300		
21	1	400	400	39	0.0975
22	2	400	400		
23	3	400	400		
24	4	400	400		
25	1	500	500	42	0.094
26	2	500	500		
27	3	500	500		
28	4	500	500		

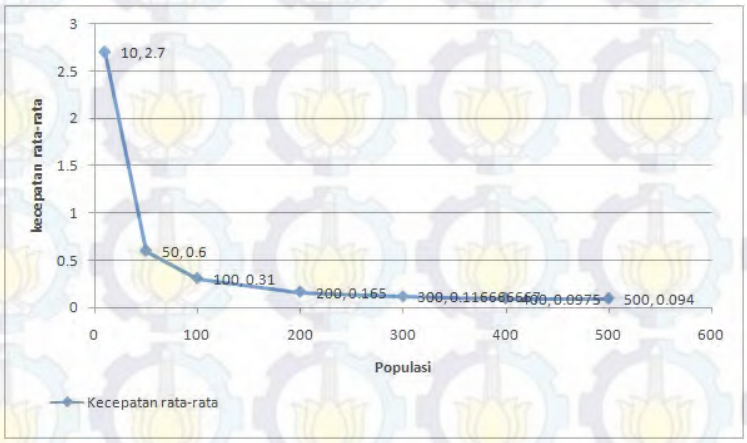
Dari data hasil pengukuran pengaruh waktu yang diperlukan untuk mencapai target utama terhadap jumlah populasi kerumunan orang pada Tabel 4.5, dapat diperlihatkan dengan grafik pada Gambar 4.6. Dari grafik tersebut terlihat bahwa, kenaikkan waktu akan terus bertambah seiring bertambahnya jumlah populasi keru-

munan penduduk untuk bergerak mencapai target utama



Gambar 4.6: Grafik Waktu Rata-Rata Pergerakan Orang Terhadap Jumlah Populasi Kerumunan

Dari data pada Tabel 4.5, dapat diperlihatkan pada grafik Gambar 4.7, pengaruh kecepatan rata-rata masing- masing orang mencapai target utama dipengaruhi dengan bertambahnya jumlah populasi di dalam area. Semakin meningkatnya jumlah populasi, akan menurunkan tingkat kecepatan rata-rata per orang untuk bergerak mencapai target utama.



Gambar 4.7: Grafik Kecepatan Rata-Rata Pergerakan Orang Terhadap Jumlah Populasi Kerumunan

4.7 Analisa Pencapaian Hasil Persentase Jumlah Penduduk yang berhasil menuju target utama

Berdasarkan Buku Masterplan Pengurangan resiko bencana tsunami (BNPB, 2012), mengenai waktu evakuasi minimal yang diperlukan oleh orang-orang yang berada dalam wilayah jangkauan tsunami jika terjadi bencana yang berpotensi tsunami

adalah dalam jangka waktu maksimal 59 menit setelah terdengarnya sirene peringatan dini. Mengacu pada hal tersebut, pada simulasi ini ditetapkan waktu evakuasi maksimal untuk para penduduk mencapai target utama jika terjadi keadaan darurat. Data hasil pengukuran diperlihatkan pada Tabel 4.5

Tabel 4.5: Pengukuran Persentasi Jumlah Penduduk Terperangkap dan Berhasil Menuju Target

Jumlah Populasi	Tidak Berhasil ke tujuan	Berhasil ke tujuan	Persentasi Berhasil
10	0	10	100
50	0	50	100
100	0	100	100
200	0	200	100
300	0	300	100
400	0	400	100
500	0	500	100

Persentase keberhasilan penduduk untuk keluar menuju target dengan jumlah populasi penduduk 10, 50, 100, 200, 300, 400 dan 500 mencapai 100 persen. Hal ini disebabkan karena waktu yang diperlukan untuk mencapai titik evakuasi atau target utama cukup lama, sedangkan jumlah populasi tidaklah terlalu banyak, yang membedakan hanyalah perbedaan waktu dari jumlah populasi yang sedikit ke populasi yang lebih banyak.

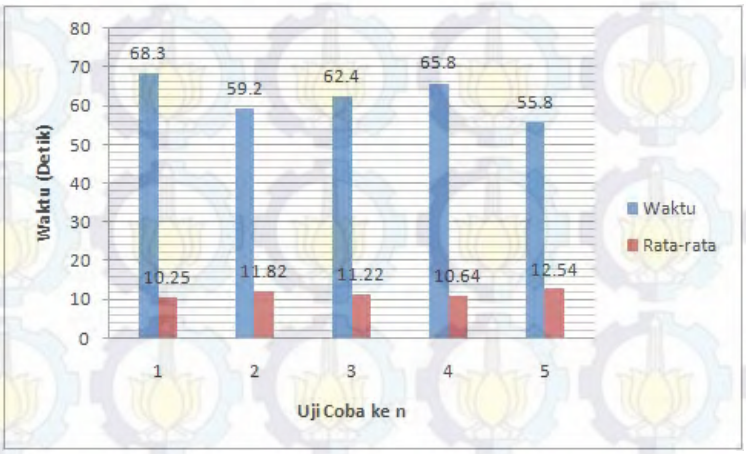
4.8 Analisa Pergerakan orang berdasarkan Kecepatan dengan Hambatan Bidang Statis

Analisa ini dilakukan untuk melakukan pengamatan pergerakan kecepatan seseorang mencapai target utama dengan menghindari hambatan bidang statis berupa bangunan dan hambatan bidang statis yang ada di tengah-tengah area simulasi. Simulasi berdasarkan variasi kecepatan berjalan. Pada simulasi ditetapkan kecepatan berjalan orang = 1,0 pixel/detik . Jumlah populasi yang dianalisa adalah 10 dengan diameter 2 pixel. Seperti terlihat pada Tabel 4.6 data hasil pengukuran simulasi variasi pergerakan kecepatan berjalan seseorang.

Tabel 4.6: Pergerakan Orang berdasarkan Kecepatan

Ujicoba ke	Waktu berjalan	Kecepatan rata-rata
1	68,3	10,25
2	59,2	11,82
3	62,4	11,22
4	65,8	10,64
5	55,8	12,54

Dari data hasil pengukuran pergerakan berdasarkan kecepatan yang diperlukan untuk mencapai target utama terhadap jumlah populasi kerumunan penduduk pada Tabel 4.6 diatas, dapat diperlihatkan secara grafik pada Gambar 4.8.



Gambar 4.8: Grafik Pergerakan berdasarkan waktu

Pada Gambar 4.8 menunjukkan bahwa, pergerakan kecepatan seseorang keluar mencapai target utama dengan menghindari hambatan bidang statis berupa bangunan dan hambatan bidang statis yang ada di tengah-tengah area simulasi adalah berbeda karena dipengaruhi oleh kecepatan awal dari masing- masing orang sesuai dengan variasi kecepatan berjalan orang tersebut. Dari grafik hasil simulasi dapat terlihat bahwa kecepatan rata-rata yang diperlukan oleh orang untuk mencapai target utama.

4.9 Analisa Kompleksitas Algoritma

Kompleksitas dari suatu algoritma merupakan ukuran seberapa banyak komputasi yang dibutuhkan algoritma tersebut untuk menyelesaikan masalah. Secara

informal, algoritma yang dapat menyelesaikan suatu permasalahan dalam waktu yang singkat memiliki kompleksitas yang rendah, sementara algoritma yang membutuhkan waktu lama untuk menyelesaikan masalahnya mempunyai kompleksitas yang tinggi.

Tabel 4.7: Analisa Kompleksitas Algoritma *Boids*

Pseudocode	Big-O
initialise_positions()	O(1)
LOOP	O(n)
draw_boids()	O(1)
move_all_boids_to_new_positions()	O(1)
END LOOP	O(n)

Pseudocode	Big-O
PROCEDURE move_all_boids_to_new_positions()	
Vector v1, v2, v3	
Boid b	
FOR EACH BOID b	O(n)
v1 = rule1(b)	O(1)
v2 = rule2(b)	O(1)
v3 = rule3(b)	O(1)
b.velocity = b.velocity + v1 + v2 + v3	O(1)
b.position = b.position + b.velocity	O(1)
END	
END PROCEDURE	O(n)

Dari tabel 4.7 dapat diketahui bahwa algoritma *Boids* merupakan Algoritma Linear. Algoritma dengan kompleksitas linear bertumbuh selaras dengan pertumbuhan ukuran data. Jika algoritma ini memerlukan 10 langkah untuk menyelesaikan kalkulasi data berukuran 10, maka ia akan memerlukan 100 langkah untuk data berukuran 100.

Tabel 4.8: Analisa Kompleksitas Algoritma A *Star*

Pseudocode	Big-O
function A*(start,goal)	
closedset := the empty set	O(1)
openset := {start}	O(1)
came_from := the empty map	O(1)
g_score[start] := 0	O(1)
// Estimated total cost from start to goal through y.	
f_score[start] := g_score[start] + heuristic_cost_estimate(start, goal)	O(1)
while openset is not empty	O(n)
current := the node in openset having the lowest f_score[] value	O(1)
if current = goal	O(1)
return reconstruct_path(came_from, goal)	O(1)
remove current from openset	O(1)
add current to closedset	O(1)
for each neighbor in neighbor_nodes(current)	O(n)
if neighbor in closedset	O(1)
continue	O(1)
tentative_g_score := g_score[current] + dist_between(current,neighbor)	O(1)
if neighbor not in openset or tentative_g_score ¡ g_score[neighbor]	O(1)
came_from[neighbor] := current	O(1)
g_score[neighbor] := tentative_g_score	O(1)
f_score[neighbor] := g_score[neighbor] + heuristic_cost_estimate(neighbor, goal)	O(1)
if neighbor not in openset	O(1)
add neighbor to openset	
return failure	$O(n^2)$

Tabel 4.9: Perhitungan *Big O* dari *function* A *Star*

Pseudocode	Big-O
function reconstruct_path(came_from,current)	
total_path := [current]	O(1)
while current in came_from:	O(n)
current := came_from[current]	O(1)
total_path.append(current)	O(1)
return total_path	O(n)

Dari tabel 4.8 dan tabel 4.9 dapat dilihat tentang perhitungan *Big O* terhadap algoritma A *Star*. adapun yang tampak pada tabel 4.8 adalah algoritma A *Star* merupakan algoritma dengan tipe polinomial yitu algoritma yang sangat kompleks, memerlukan waktu yang cukup banyak karena memerlukan jumlah langkah penyelesaian yang lebih banyak dari pada data yang dimasukkan.

[*Halaman ini sengaja dikosongkan.*]

BAB 5

PENUTUP

5.1 Kesimpulan

Dari penelitian yang dilakukan dengan menggunakan metode *boids* dan A *star* dengan melakukan simulasi kerumunan untuk pergerakan penduduk menuju titik evakuasi, dapat disimpulkan sebagai berikut :

1. Penggunaan algoritma *boids* dan A *star pathfinding* berhasil diterapkan sebagai algoritma untuk simulasi kerumunan menuju target tertentu.

2. Meningkatnya jumlah populasi akan menurunkan tingkat kecepatan rata-rata penduduk untuk mencapai target . Dengan jumlah populasi awal 10 kemudian ditingkatkan menjadi 50 kali, kecepatan rata-rata mengalami penurunan sebesar 96,52%.

3. Meningkatkan waktu yang diperlukan oleh penduduk untuk mencapai target dengan semakin meningkatnya populasi. Dengan jumlah populasi awal 10 kemudian ditingkatkan menjadi 50 kalinya, maka waktu yang diperlukan mengalami peningkatan sebesar 35,71%.

4. Algoritma *boids* dan A *star* memerlukan waktu untuk melakukan evakuasi yaitu rata-rata sebesar 42 menit. Waktu rata-rata ini tidak melampaui waktu maksimal yang ditetapkan oleh BNPB untuk melakukan evakuasi tsunami yaitu kurang dari 59 menit.

5. Simulasi evakuasi tsunami yang memakai desain perkampungan, dengan rumah-rumah sebagai hambatan statis menunjukkan tingkat keberhasilan evakuasi yang ditandai dengan tercapainya 2 faktor keberhasilan evakuasi yaitu waktu evakuasi yang tidak melampaui waktu maksimal dan prosentase jumlah penduduk yang selamat mencapai 100%

5.2 Penelitian Selanjutnya

Penelitian ini bisa dikembangkan untuk permasalahan simulasi kerumunan pengunjung dengan menerapkan pada environment lebih dari komplek dan dengan titik evakuasi lebih dari satu. Dan hambatan dinamis dapat di tambahkan lagi dengan hambatan dinamis seperti kobaran api, reruntuhan benda, mobil atau

kendaraan lain yang ikut bergerak dan lain-lain, sehingga simulasi dapat lebih mewakili keadaan sebenarnya didalam dunia nyata. Penelitian ini dapat dikembangkan dengan menerapkan pada simulasi 3 dimensi dan dengan memberikan perilaku kecerdasan buatan lainya pada objek, agar objek orang atau pengunjung mempunyai perilaku yang lebih natural. Untuk itu perlu dilakukan penelitian lebih lanjut untuk mencapainya.

Daftar Pustaka

Abrahams, John, 1994 , *Fire escape in difficult circumstances* ,chapter 6, In: Stollard.

Aspelin, K, 2005, *Establishing Pedestrian Walking Speeds*, Karen Aspelin, P.E., P.T.O.E., ITE District 6 Technical Chair Parsons Brinckerhoff Albuquerque, New Mexico. Portland State University

Compsci, 2006, *Collision Detection Tutorial*, at http://compsci.ca/v3/ viewtopic.php?t=13661

Cui, X, 2006, *A Flocking Based Algorithm for Document Clustering Analysis*, Journal of System Architecture

Dewi, Meilany, 2011, *Simulating The Movement Of The Crowd In An Environment Using Flocking*, Journal of 2011 International Conference on Instrumentation, Communication, Information Technology and Biomedical Engineering, Bandung

Dewi, Meilany, 2012, *Simulasi Pergerakan Pengunjung Mall Menggunakan Flocking dan Obstacle Avoidance*, Institut Teknologi Sepuluh Nopember, Tesis

Kurniawati, A, 2010, *Simulasi Pergerakan Pengunjung Mall Menggunakan Potential Field*,Institut Teknologi Sepuluh Nopember, Tesis

Maulana, S, N, 2010, *Penggunaan Struktur Data Quad-Tree dalam Algoritma Collision Detection pada Vertical Shooter Game*,Makalah IF3051 Strategi Algoritma Sem. I Tahun 2010/2011, Institut Teknologi Bandung, Bandung

PIGE, 2001,*Collision Detection*, at http://www.edenwaith.com/products/pige/ tutorials/collision.php

Reynolds, C.W, 2010, *Steering Behaviors For Autonomous Characters*, http://www.red3d.com/cwr/steer/gdc99/

Stuart Russell, Peter Norvig, 1995, *Artificial intelligence : a modern approach* Prentice-Hall, Inc. A Simon and Schuster Company, Englewood Cliffs, New Jersey 07632

Undang-undang Nomor 24 Tahun 2007, *Penggulangan Bencana*.

Yozo Goto, Muzailin A, Agussabti, Yudha Nurdin, Diyah K. Yuliana, Ardiansyah, 2012, *Tsunami Evacuation Simulation for Disaster Education an City Planning*,Journal of Disaster Research Vol. 7 No. 1, 2012

Wikipedia, 2014, *A* search algorithm*, at http://en.wikipedia.org/wiki/A*_search_algorithm

BIOGRAFI PENULIS

Nama : I Made Pasek Mudhana
TTL : Klungkung, 28 Desember 1980
Agama : Hindu
Alamat I : Jl. Laksamana 42 b, Singaraja- Bali
Alamat II : Jl. Hidrodinamika III, T88 Sukolilo, Surabya
HP : 085253708526
Email : madepasek17@gmail.com

Jenjang Pendidikan :

1. Tahun 1987 - 1993 : SDN 1 Baktiseraga, Singaraja - Bali
2. Tahun 1993 - 1996 : SMP Negeri 2 Singaraja
3. Tahun 1996 - 1999 : SMA Negeri 4 Singaraja
4. Tahun 1997 - 2002 : IKIPN Singaraja
   Fakultas Pendidikan Teknologi Kejuruan
   Program Studi D3 Manajemen Informatika
5. Tahun 2007 - 2009 : STMIK Denpasar
   Program Studi Teknik Informatika
6. Tahun 2013 - 2015 : Institut Teknologi Sepuluh November
   Fakultas Teknologi Industri
   Jurusan Teknik Elektro
   Program Studi Telematika
   Konsentrasi *Chief Information Officer*

Riwayat Pekerjaan :

1. GTT di SMAN 3 Singaraja (2002 – 2006)
2. Bappeda Kab. Karanagsem - Bali (2006 – 2011)
3. Diskominfo Kab. Karangasem - Bali (2011 – 2013)
4. BPBD Kab. Karangasem - Bali (2013 – Sekarang).